Introducing distributed dynamic data-intensive (D3) science: Understanding applications and infrastructure

Citation for published version:
Jha, S, Katz, DS, Luckow, A, Chue Hong, N, Rana, O & Simmhan, Y 2017, 'Introducing distributed dynamic data-intensive (D3) science: Understanding applications and infrastructure' Concurrency and Computation: Practice and Experience, vol 29, no. 8, e4032. DOI: 10.1002/cpe.4032

Digital Object Identifier (DOI):
10.1002/cpe.4032

Published In:
Concurrency and Computation: Practice and Experience

Introducing Distributed Dynamic Data-intensive (D3) Science: Understanding Applications and Infrastructure

Shantenu Jha[1*] Daniel S. Katz[2], Andre Luckow[1],
Omer Rana[3], Yogesh Simmhan[4], Neil Chue Hong[5]

[1] *Rutgers University*
[2] *University of Chicago & Argonne National Laboratory*
[3] *Cardiff University*
[4] *Indian Institute of Science*
[5] *University of Edinburgh*

SUMMARY

A common feature across many science and engineering applications is the amount and diversity of data and computation that must be integrated to yield insights. Data sets are growing larger and becoming distributed; and their location, availability and properties are often time-dependent. Collectively, these characteristics give rise to dynamic distributed data-intensive applications. While "static" data applications have received significant attention, the characteristics, requirements, and software systems for the analysis of large volumes of dynamic, distributed data, and data-intensive applications have received relatively less attention. This paper surveys several representative dynamic distributed data-intensive application scenarios, provides a common conceptual framework to understand them, and examines the infrastructure used in support of applications.

KEY WORDS: dynamic, distributed, data-intensive, scientific applications

1. INTRODUCTION: CONTEXT, SCOPE AND OUTLINE

The landscape of scientific and enterprise computing is being fundamentally altered by prodigious volumes of data. This massive data generation can be naturally distributed because they are being created in a set of distributed locations, rather than in one place. At the same time, data can also be transported to be processed in multiple, distributed locations, rather than being processed near their source, or be brought together for centrally processing at a particular location. These two orthogonal dimensions lead to a number of patterns of data generation and processing being present in real applications, either for reasons of performance or for collaborative analytics. Further, the variation of data production rates, data source, and destination makes temporal variations in data properties increasingly important. With an increase in the importance of distributed and temporal properties of data, systems and infrastructure aspects such as resource scheduling, data placement, and transfer decisions that were statically determinable at small scales emerge as dynamic and important considerations at large scales of data and distribution.

The focus of this paper is on a subspace of the large set of problems associated with data-intensive sciences, namely that which is characterized by *both* distributed and dynamic aspects.

*Correspondence to: Rutgers University, NJ, USA. E-mail: shantenu.jha@rutgers.edu

We introduce the concept of Dynamic Distributed Data-intensive science and refer to this subspace of data-intensive science as D3 science.

The analytical space spanned by the three "D"s of dynamic, distributed, and data-intensive provides a common vocabulary and effective framework to analyze applications and infrastructure alike. Subjecting a range of otherwise distinct applications to a common analytical framework provides an opportunity to search for similarities across applications and understand core differences. The diverse D3 scenarios point to the need for a careful understanding of D3 characteristics as well as sophisticated, extensible, and skillfully architected infrastructure. A common framework thus informs infrastructure scientists and cyberinfrastructure experts where effort is likely to yield maximum returns. The advantages of providing a common but extensible vocabulary that can be used by the entire community—application end-users, developers and infrastructure scientists alike—in an emerging field cannot be overstated. We believe that such a vocabulary and terminology has been missing from the practice of distributed cyberinfrastructure in general, with negative consequences.

1.1. Scope

One motivation of this work is to understand the dynamic properties and the distribution of applications, and the complex interplay between them that arises as a consequence of scale. Additionally, we are motivated by an attempt to understand infrastructure—existing capabilities, trends, and limitations—available to support the complexity, challenges and characteristics of dynamic and distributed data at scale.

We would like the reader to keep in mind some points regarding context and scope:

- This *survey* is not meant to be complete, either in the scope of domains covered or the types of infrastructure and application surveyed. It represent the interests and experience of the authors. The specific requirements and barriers of applications cited in this paper are representative of the time at which the applications were surveyed (2012-2013); this is also true of the quantitative analysis of applications.

- The focus is generally on applications and infrastructure that arise from science and engineering projects in academia in general; this is a guiding principle. While we are cognizant of enterprise applications and infrastructure, and recognize the impressive advances made therein, we also acknowledge that the factors such as scale, distribution, open-source, and interoperability influence the solutions employed and infrastructures used; it is important to therefore focus on problems that emerge in the space defined by these "academic constraints".

- Almost by definition, any attempt to capture the state-of-the-art of such a fast-changing and emerging domain is bound to be fraught with limitations and issues of scoping. A complete and rigorous survey is bound to be obsolete by the time it emerges; so given the rate of change, we asked what could be done. What we present is a partial analysis motivated by a set of existing and active D3 applications, their trials and tribulations, successes and solutions. We believe there is merit in capturing the state-of-play as is, deriving a research agenda and set of specific yet crosscutting issues, and presenting these to the community.

- Distribution can occur on many levels. In this paper, we use *locally distributed* to indicate that data generation and storage, or computation over it, occurs on multiple nodes within one data center. *Geographically distributed* describes the distribution of data generation, storage and/or compute across multiple data centers. Also, our focus is primarily on infrastructure considerations at large scale, as searching for common solutions and patterns at large scale is likely to be more fruitful than at small scales where customized tools and optimization of low-level systems features is likely to return greater yield.

- We use the term dynamism to refer to the spatio-temporal variability of data (e. g. data rates and formats), applications (e. g., workflows that depend on data values) and infrastructures (e. g. availability).

1.2. Outline of this paper

The remainder of this paper consists of the application scenarios (§2), a discussion of the aspects of distribution and dynamism in the applications (§3), a description of the software infrastructure required for D3 applications (§4), analysis and characterization of the applications and conclusions (§5). Each of these sections is briefly described below:

- Section 2 provides a description of thirteen D3 applications based on our survey. For each application, both the *dynamic* and *distributed* aspects are described. The methodology used to carry out the survey is described in the Appendix.

- Based on the application survey, we derive a set of definitions for *Distribution* and *Dynamism* in §3. Note that these definitions are based on the application survey. A variety of definitions of these concepts already exist in computer science literature. Our intention is to provide context for the use of these terms with reference to our survey. The distributed and dynamic characteristics of each application surveyed in §2 is summarized in this section.

- In §4, we subsequently describe the software/systems infrastructure used to support the distributed and dynamic characteristics of applications in §2. We outline support for data management, analysis, and coordination, providing examples of actual cyberinfrastructure made use of by our surveyed applications for these purposes.

- In §5, we bring together key observations based on the analysis of applications, their distributed and dynamic characteristics, and infrastructure use. An attempt is made to compare these applications and suggest the need for an "architecture for D3 science".

Guidance for the reader: Readers who are generally knowledge about distributed applications will likely feel comfortable reading the paper from start to end. Readers with less distributed applications knowledge may prefer to read §3 and §4 before §2, or might want to read §2 first, but jump to §3 and §4 when they find terms which which they are unfamiliar. Unfortunately, we do not feel that there is a single ordering of the information in this paper that is best for all readers.

2. D3 APPLICATION SCENARIOS

Starting with the experience of the attendees of the Dynamic Distributed Data Programming Abstractions and Systems (3DPAS) workshops, we have assembled a set of D3 "applications" that we describe in this section.

The term "application" is often overloaded in literature and the computational science discourse. Sometimes the term is a reference to a standalone, independent executable (e.g., a Molecular Dynamics simulation kernel such as AMBER or Gromacs). Sometimes it refers to science problem to which the executable is put to use (e.g., computing the free-energy of binding energy of a drug candidate), but sometimes it is also used to reference the end-to-end workflow (i.e., a multi-stage computational process requiring a sequence of possibly distinct simulations and executables). To add chaos to the confusion, applications are sometimes also as a proxy for projects (e.g., the ATLAS project), where the project could itself possibly be a series of independent applications, with each application in turn a reference to a defined science objective achieved using a set of executables.

In addition to the aforementioned hierarchy and granularity to which the term application has been applied, the class of applications in any level differ widely in scale, sophistication and type as well different objective and objects. Thus, we warn the reader of the inevitable befuddlement that the casual and non-contextual use of the word application can cause.

Against this backdrop, we eschew a formal classification of application types, however, to facilitate understanding we categorize applications into two categories. The first is that of *traditional applications*, where a program (or set of programs) is developed independently by a user or project to try to answer a science question (or set of related questions). A single standalone computer program, a distributed application, or a complex workflow could fall under the category of a *traditional application*. Applications in this paper that are in this category are NGS (next generation sequencing) Analytics (§2.1), CMB (cosmic microwave background) (§2.5), Fusion (§2.10), Industrial Incident Notification and Response (§2.11), MODIS (moderate resolution imaging spectroradiometer) Data Processing (§2.12), and Distributed Network Intrusion Detection (§2.13).

Another group of applications are *infrastructural applications*. Applications of this type are collaborative, in that they depend on a context that is often agreed upon by a community, most often in terms of how data is stored by the community and how it is accessed, as well as, more often than not, a weak consensus on the infrastructure to be used. This allows different sub-applications (which themselves are full-fledged "applications", but with dependencies on the collaborative infrastructure) to focus on different stages, such as generating data, or processing data. Answering a science question now may involve a set of applications that need to be run in series, perhaps in different stages, that may be run by different groups that do not frequently interact. In this paper, examples of this type of application include ATLAS/WLCG (a toroidal large hadron collider apparatus/worldwide large hadron collider computing grid, §2.2), LSST (large synoptic survey telescope, §2.3), SOA (service-oriented architecture) Astronomy (§2.4), Sensor Network Application (§2.6), Climate (§2.7), Interactive Exploration of Environmental Data (§2.8), and Power Grids (§2.9).

In other words, multiple infrastructural applications are used together in particular science domains. Furthermore, infrastructural applications involve more than just the end user (scientist), and typically require exerting control at lower layers, i.e., "programming the system (infrastructure)", as opposed to a traditional applications where end user effort and control is often confined to a well-defined application kernel and their communication and coordination. Also, for some *traditional applications* (e.g., NGS) the number of users of an instance of the application is typically small, i.e., the predominant use is by the long-tail of science. In contrast, some infrastructural applications are correlated with "big science" projects.

Some of the applications we describe are really groups of applications, meaning that we are grouping together a number of independent programs, written by different authors, that are trying to answer the same type of science question, where a single user will only run one of them at a time. These codes and their developers may either be competing or collaborating, and sometimes both, informally known as collabetition. Specifically, this is the case for NGS Analytics (§2.1), SOA Astronomy (§2.4), and Interactive Exploration of Environmental Data (§2.8). (Note that this issue is orthogonal to traditional vs. infrastructural applications: NGS Analytics is traditional, where SOA Astronomy and Interactive Exploration of Environmental Data are infrastructural.) In each of these examples, we have chosen one specific application that is meant to represent a possibly large number of comparable applications.

The discussions in the workshops and our analysis of the applications led us to examine the use of data in different parts of the applications, and informed the terminology we use in the rest of the paper. We have split the presentation of information on each scenario into three aspects: *big data aspects*, concerned with the number and volume of data; *distributed aspects*, concerned with the way data generation, storage and processing are separated; and *dynamic aspects*, concerned with the spatio-temporal variability of the data or application. Doing this has enabled us to identify some common patterns across applications. For example, applications that involve streaming data collect data from sensors, transform and filter the data, store the results, and then later process (analyze) the stored results. In general, we think that all of the applications we have studied can be mapped to a set of stages or phases. This is illustrated in Figure 1.

The question of where the application, as opposed to the entire system, actually starts is somewhat difficult. In this paper, we have decided not to treat hardware data sources (e.g., sensors, telescopes) as part of the applications. Therefore, these applications generally start with the data movement that follows the actual data generation.

Unlike the sensor applications, another set of applications generate data computationally, for example, from a model or simulation. These applications then do start with data generation, and they often have just three stages, where data is generated in the first stage, stored in a second stage, and then processed in a third stage (i.e., the transformation & decisions stage in Figure 1 does not occur).

Other applications may operate on stored data (i.e., they only contain the processing stage and its associate data movement).

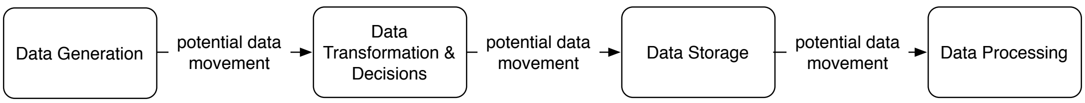

Figure 1. Application Stages: The work in data-intensive applications can be put into four common application stages: data generation, transformation & decisions, storage, and processing, though not all applications have all stages.

2.1. *Next Generation Sequencing (NGS) Analytics*

The (grand) challenge in bioinformatics is the need to provide support for the analysis of genome sequence data that is becoming available due to the abundance and high-throughput capability of the next-generation sequencing devices. The rate of growth of data from the NGS machines humbles Moore's law; it has increased by more than three orders of magnitude in barely a decade.

There are many different types and components of analyses of NGS data. However, fundamentally important to all types is the ability to efficiently and effectively map/align the short reads to a reference genome.

Alignment underpins multiple scientific questions such as comparing metagenomic (DNA sequence) data sets against each other, which could be used to distinguish human disease biomarkers among different individuals, determine different soil (or water or air or ...) microbial composition from different samples, etc.

This subsection describes a set of *traditional applications* that are used in the processing of next generation sequencing data. The applications have two of the possible stages: data storage and data processing.

Application description There are multiple tools (programs) that can perform the mapping/alignment phase, e.g., BWA, BFAST, Bowtie, MAQ, etc. Each has its own strength and its own native data support/capability; they also implement different alignment algorithms. Thus, based upon different design objectives and trade-offs, they typically have different constraints and efficiencies.

Several of these tools are also composed together as a linear pipeline for performing sequencing and analysis. The pipeline may go through alignment, SNP calling, haplotyping, and so on, with different tools used in each stage. These pipelines can often be performed independently on each chromosome to provide simple parallelism.

Some NGS analytics are not good fits for core-rich and memory-poor modern architectures. Some problems (e.g., comparing k-mers across metagenomic data sets) need terabyte-scale memory, not thousands of cores. [1]

Individual data sets, which will be terabyte-scale, are generated by NGS machines at a variety of locations. There could be many thousands of such sets to compare. Having them all local may not be feasible (the transfer time from where they are generated may be prohibitive).

Big data aspects Individual data sets will be terabyte-scale. For example, for a standard genomic sequence, there are billions of short (35-100 bases) reads of DNA. And there could be many thousands of such sets to compare for a given problem. Getting more accuracy for sequencing may require multiple coverage (e.g. 10x), that can cause sequence sizes to increase linearly. This makes NGS data management fundamentally a big data problem. NGS analytics, in addition to being a big data problem, is also a computationally demanding and distributed computing problem. The computational demands arise from the often complex and intensive analysis that has to be performed on data, which in turn arise from algorithms that are designed to account for repetitions, errors and incomplete information.

Distributed aspects The type of NGS research being conducted can distinguish the distributed properties of the application. Researcher groups that own sequencing machines need to deal with large dataset sizes that need to be re-sequenced and reduced for further analysis. Other researchers may analyze large catalogs of sequenced genome databases are maintained, and perform pattern matching. Further, clinical may be interested in in-depth analysis of an individual sequence for disease diagnosis or visualization.

The distributed computing aspects arise at multiple levels: for example, the simple act of having to move data from source (generation) to the destination where computing (analysis) will occur is a challenge due to the large volumes involved. Trade-offs exists between the cost/challenges in distributing data versus I/O saturation or memory bottlenecks.

When coupled with the compute intensive nature of the problem, it soon emerges that a fundamental challenge is not only whether to distribute, but where to distribute, what to distribute (should the computing move to the data, or the data move to the compute), and how to distribute (what tools and infrastructure to use).

Dynamic aspects The dynamic challenges of NGS data are more subtle; the data themselves are not dynamic – the sequences are acquired once and analyzed many times. However the execution of these applications on distributed infrastructure have dynamic aspects when optimized resource usage is considered. For example, workload decomposition and distribution must be determined dynamically in order to optimally use resources, e.g., compute resource selection can be based on data location, processing profile and/or network capacity.

Other important issues There are several toolkits that have emerged for NGS providing incrementally better algorithms for sequencing and analysis [2, 3, 4, 5]. However, as a consequence of the broad range of infrastructural requirements, there isn't currently a common "standard" cyberinfrastructure used for NGS analytics; researchers use what is easily available to them, or try to use COTS infrastructure. Historically, leaders in the field have developed infrastructure for the larger community to build-upon, e.g., QCDOC or MD-GRAPE. But given the broad range of requirements, a hardware-only solution is unlikely for a broad range of problem instances, and a combination of hardware and software approaches will be employed. This reinforces the need for flexible programming systems and associated run-time environments that support collective (hardware-software) optimization.

NGS sequencers such Illumina and Ion Torrent provide the option of onsite servers or small clusters for performing the alignment and re-sequencing as a pipeline right after the reads have been generated. Scientific workflow systems like Taverna [6], BioKepler [7], and Galaxy [8] (though Galaxy does not support distributed systems currently) have been successful in integrating genome analysis tools for bioinformatics. Script-based pipelines that combine executables are also often used. Very few workflow systems have explicit support for distributed data or dynamic data handling. Therefore reusable approaches at extending existing tools to support distributed and dynamic data are likely to be useful.

2.2. *ATLAS (an Example of Use of the WLCG)*

The Large Hadron Collider (LHC) [9] at CERN in Geneva is producing a large amount of data across a number of experiments, including ATLAS [10]. The application discussed in this section is an *infrastructural application* (meaning that it consists of multiple stages that are run by different people, in this case data acquisition, storage, distribution, and analysis). It has the goal of allowing a physics group or a user to analyze data to understand a specific physics channel (underlying physical process) as recorded by ATLAS. The set of available data grows steadily over time as more data are collected (by the experiment) and reconstructed. This application thus contains two stages: data storage and data processing.

Application description Both physics groups and individuals run jobs on the WLCG (Worldwide LHC Computing Grid) [11] as they see fit, either reading the reconstructed data or reading intermediate datasets created by themselves or their colleagues. If, in the future, resources should prove inadequate then adjustments to working practices to coordinate processing may be made to increase physics output.

The WLCG has a hierarchical architecture. CERN (the ‘Tier 0’ site) is connected to eleven ‘Tier 1’ (national) sites by a dedicated optical network. This may be extended in the future. All data is stored at CERN, and as needed, various parts of the dataset are copied (and cached, i.e., replicated) to the Tier 1 sites, and from there to ‘Tier 2’ (regional) sites and to ‘Tier 3’ (university or group or individual) sites.

Before data is analyzed, the file to be analyzed is copied from the closest site to the SE (Storage Element) at the WLCG site where the job will run. From there it is either copied to a local disk or read from the SE.

There is no particular time constraint on the processing, although physics groups are eager to understand the ATLAS experiments and thus want to process data as quickly as possible. In the event some data are missed during processing, the only effect is a reduction of statistical precision in the final analysis. Accurate bookkeeping is therefore important in order to know which data have been processed.

Big data aspects Roughly 20 TB of data are collected every day. Multiple generations of processed data are kept. Many data files are replicated on multiple sites; this is done both by copying it to where it is likely to be useful and partly dynamically as the need arises. The bulk of the data is individual event data at different stages of refinement. This is held in an in-house representation of C++ serialized objects. Though this format is not standard it is used by all LHC experiments.

Distributed aspects The WLCG has 250,000 cores distributed over 140 sites and with 100 PB of disk. It uses a mixture of gLite software, experiment specific software and physics group specific software. The data is distributed and replicated as described previously.

Dynamic aspects The data being processed can be considered as dynamic as it grows steadily as new raw data are collected and the basic processing program (called reconstruction) is run to produce information about specific particles involved in a collision. This reconstruction is rerun periodically (two or three times per year is expected) as calibrations of the detector are improved. Reconstruction algorithms are also improved as are the physics groups’ selection codes.

Other important issues Data analysis is really a pleasingly parallel task, where various events are analyzed in parallel and then a summary is created of all the analyses.

The key challenge here is really in the infrastructure – building a system that can store massive amounts of data and move them to where the processing is to be done by a large diverse group of users.

2.3. LSST

This application, which similarly to the previous application is an *infrastructural application*, wants to find and study variable objects and moving objects, using a telescope that takes images each night. This is a scenario that may be used by Large Synoptic Survey Telescope (LSST) [12], and is somewhat similar to that done by the Palomar Transient Factory (PTF) [13, 14, 15] and by Pan-STARRS [16, 17]. Overall, this application is a leading-edge example of the work needed to build a set of data (catalog and images) that then can be used by others for analysis (an example of which is the next application, in §2.4.) This application contains three stages: data transformation & decisions, data storage. (Data generation comes from the telescope itself, and processing is done outside of this application, e.g. using the next application.)

Application description Processing of data (data analysis and detection of new sources) from each image should be done while the next image is being taken by the telescope. This means that if anything unusual is detected, normal observation can be interrupted. Other observing resources can also be notified instantly, so they can observe the same event. As data is collected, it is added to all the data previously detected from the same location of sky to create a very deep master image. LSST will also build up a database of all known moving objects.

Every time a new image of the sky is obtained, the master image will be subtracted from it. The result is a subtracted image that contains the difference between the current sky image and its average state.

This subtracted image is then processed by a cluster of computers with 3 main steps:

(i) Using the existing object catalog (containing the known orbits of all known objects), find objects which are expected to appear in subtracted image, given the area of sky and time of day. Cross-match expected objects with sources in subtracted image, resulting in the unmatched-source catalog, which contains sources that cannot be matched with a previously known object. If these sources can be tracked over a few images, an orbit can be determined for them and they can be added to the object catalog. Also, the orbit catalog can be updated with re-detections of known objects; each rediscovery provides information that improves the known orbits.

(ii) Attempt to classify all entries in the orbit catalog (i.e., all the objects which now have known orbits). If any Near Earth Objects are detected to be passing close to the earth, alerts are generated to astronomers so that follow up observations can be scheduled.

(iii) If an unknown object is found that cannot be classified locally, at least two possible options exist. One is a coordinated effort of comparison with other observatories, where the local observatory issues a call for participation to a set of additional possible observatories, each of which can accept or reject the call. The local observatory then chooses one of the accepting observatories from which to obtain data, and that observatory takes data and returns it. The local observatory then decides what to do: classify the object, contact a human, or call for more observations (automatically to other observatories that decide whether or not to accept, and when)

A second option is less coordinated, where this observatory would send VOEvent messages, notifications of transient events which can be used to trigger follow-up observations on other telescopes. These are small XML messages containing basic information about the position of a transient source and a small amount of other metadata which the operators of a given telescope can use to define triggers for follow-up observations of interest to them or their community.

Big data aspects LSST will generate 36 GB of data every 30 seconds, and over a 10-hour winter night, will collect up to 30 TB. Data will be stored as FITS image, and in large databases containing analyses of the discoveries.

Distributed aspects The overall system will contain a central telescope with computing data and resources, linked to a wide area network of observatories and data and computing resources.

Dynamic aspects The time constraints differ for different types of transient sources, but in some cases notification of remote telescopes must be made within a few minutes if it is to be useful; the overheads associated with setting up a follow-up observation mean that much shorter time scales are practically irrelevant, although they are still scientifically interesting. The quality constraints center on providing a remote telescope with good enough information about a potential transient source that it can make a reliable decision about whether to interrupt its existing observing program to follow-up the event. In most cases, the candidate transient will be some sort of noise event, so fairly sophisticated filtering is required to bring the false positive rate down to something acceptable, given the cost of observing time on cutting edge telescopes.

Other important issues Data will initially be analyzed at the telescope; discoveries will be sent to other centers.

Pan-STARRS, a contemporary sky survey with similar goals as LSST, uses a scripted programming model on a cluster for image processing of telescope frames to extract object attributes as a CSV file. This is followed by followed by scientific workflows, based on Trident, that load the files into individual databases daily, and merges them with a master distributed-database on a weekly basis on an HPC cluster [18]. Data extraction and mining operations by astronomers are composed using a pseudo-SQL language with user defined functions and submitted for execution on the master database using a batch system.

2.4. SOA Astronomy

An increasingly important feature of astronomical research is the existence of systematic sky surveys covering the whole sky (or large fractions thereof) in many different wavebands. The principal outcome from these surveys are astronomical source catalogues, which contain values for attributes characterizing all the celestial sources detected in the sky survey dataset. These are now typically implemented in relational databases, with SQL interfaces implemented through web pages, and the principal goal of the Virtual Observatory (VO) being developed by the International Virtual Observatory Alliance (IVOA) [19] is to standardize access to these, and other astronomical data resources, so that astronomers can readily perform multi-wavelength analyses, combining all the extant data on particular sources. The previous application (§2.3) was an example of a “generator” of such data resources, this application is a typical “user” of them. As such, it can generally be considered as having just the last application stage: data processing. Thus it is an *infrastructural application*. The particular example described here calculates the photometric redshift for a given area of sky using two tools: HyperZ and ANNz, both accessible via web services interfaces [20]. However, it represents many service-oriented architecture applications that are designed to work with the VO.

Application description The VO includes a registry that contains metadata describing all the data resources published using VO data access standards, enabling the discovery of datasets relevant to a particular analysis. Web service implementations of data access protocols enable users to access data in these repositories programmatically, and distributed scratch space is provided for the storage of intermediate result sets, so the user does not need to route large data flows through his/her own workstation. In the future, more and more data analysis software will be made available through web services compliant with VO standards, enabling more of the data integration and analysis process to be combined in workflow.

The application orchestrates a set of services through a pipeline in order to compute the redshift of a given area of sky. It contains a set of components. 1) The Wide Field Survey Archive (WFS), which is an image catalog. 2) A spectroscopic database, used to provide spectroscopic data for an area of sky. 3) The SExtractor tool, which is an image processing that extracts all objects of interest (stars, galaxies, etc.) 4) A cross matching tool that compiles one table of all objects of interest over the five wavebands. 5) Hyperz, the first photometric redshift estimation algorithm. 6) ANNz, the second photometric redshift estimation algorithm, and 7) mySpace, the AstroGrid storage service.

Big data aspects Images will be O(1) GB files.

Distributed aspects The data resources published to the VO exist in a number of data centers distributed internationally. Major datasets—e.g. those from large sky surveys—are typically implemented in relational databases running on high-spec servers or clusters thereof. The services in this application are also distributed, with web service interfaces used to link them.

Dynamic aspects The WFS catalog will be updated regularly, therefore a query at time x will not provide the same data as a query at time y. In addition, the specific queries that will be issued by users are unknown, and will change over time.

Other important issues Enough data needs to be available for a given area of sky from both the WFS archive and the spectroscopic archive.

2.5. *Understand the Cosmic Microwave Background*

This *traditional application* performs data simulation and analysis to understand the Cosmic Microwave Background (CMB) [21], which is an image of the Universe as it was 400,000 years after the Big Bang. Tiny fluctuations in the CMB temperature and polarization encode the fundamental parameters of cosmology and, using the Big Bang as the ultimate particle accelerator, ultra-high energy physics. Extracting this information from the data gathered by current and anticipated CMB observations is an extremely computationally intensive endeavor for which massively parallel tools have been developed. This application contains just one stage: data processing.

Application description The CMB community wants to extract cosmology and fundamental physics from the observations gathered by the detectors as time-ordered sequences of O(10^{12} - 10^{15}). These observations are reduced first to a map of O(10^6 - 10^8) sky pixels, then to O(10^3 - 10^4) angular power spectrum coefficients, and finally to O(10) cosmological parameters [22].

The central computing challenge for any CMB dataset is the simulation and analysis of O(10^4) synthetic observations, used to correct for biases and quantify uncertainties in the analysis of the real data. Preconditioned conjugate gradient techniques are used to solve for the maximum likelihood sky map given the input data (obtained by scanning the sky with hundreds to thousands of detectors for weeks to years) and its piecewise stationary noise statistics. To avoid the I/O bottleneck inherent in the traditional simulate/write/read/map paradigm, all simulations are performed on the fly only when requested by the map-making code.

Going from the map to the angular power spectrum is the computationally most expensive step. The exact solution scales with the cube of the number of pixels (in the map), so going from map to angular power spectrum is ruled out now (but was possible earlier for much smaller observations). The approximate solution requires sets of O(10^4) Monte Carlo realizations of the observed sky to remove biases and quantify uncertainties, each of which involves simulating and mapping the time-ordered data.

The map-making application is therefore applied to both real and simulated data, but many more times to simulated data (which requires us to use the on-the-fly simulation module too).

Big data aspects There is O(1 - 10) TB input data and a similar volume of output data.

Distributed aspects This application is targeting the largest supercomputers available: Hopper, Tianhe, Blue Waters, etc. It is all about the cycles, although as the concurrency increases previously solved I/O and communication scaling bottlenecks typically re-emerge.

The application currently uses a single HPC systems, but the scientists have discussed using distributed systems, with a model of remote systems being used for the data simulations. Each simulation would be launched from the central system that is building the map, and output data from the simulations would be asynchronously delivered back to that central system as files that would be incorporated in the map as they are produced.

Dynamic aspects Overall, this application can make use of whatever computing it can access in order to run the simulations needed to flesh out the observed data (which can be read from local disk). The simulations are dynamic in time and in location.

Other important issues A goal is to simulate and map $O(10^2)$ realizations of an experiment's data in O(1) wallclock hour to provide 10% errors during the early analysis stages, and $O(10^4)$ realizations in O(100) wallclock hours for the final definitive analysis.

2.6. *Sensor Network Application*

Marine sensing covers a number of applications, including environmental monitoring, remote exploration, marine life surveys, and habitat assessment. In each case, the main challenge is collecting data reliably from a difficult working environment, without interfering with the natural behaviors the animals exhibit.

A canonical example is the monitoring of seal and other sea mammal populations by the Scottish Oceans Institute (SOI), which tags animals with sensor packages that can record dive behavior, speed, and movement [23, 24]. These datasets are then analyzed offline using traditional statistical techniques and integrated with Google Earth to visualize animal tracks on a large scale. This is an *infrastructural application*. It begins after data generation, and it includes the data storage and processing stages.

Application description The sensor packages on the marine mammals report back to base when the animal crawls up a beach and comes within range of a cellular network. Data are collected remotely and brought to a central site for analysis. Analysis includes statistical applications and visualization is done using Google Earth.

Big data aspects The data sets collected are modest in size by scientific-data standards. They include positions and motion vectors for sea mammals, and concentrations and gradients for environmental missions.

Distributed aspects Time series data is published from distributed sensors, but they are stored in a central repository. The data is analyzed locally and visualized by distributed users.

Dynamic aspects Time series data is published dynamically from the sensors when they are within communication distance of cellular towers. So the time series data arrives at the central repository asynchronously relative to the time of collection.

There is some basic adaptation to the resolution of the data collected to account for different resolutions of the sensor packs. Larger-scale environmental sensing applications may make use of more structured and extensive adaptation, such as changing the sampling frequency and other management characteristics in response to the data being observed.

Other important issues Collecting data reliably is challenging due to the distributed deployment of sensors in hostile environments.

2.7. Climate

This *infrastructural application* is aimed at supporting international CMIP/IPCC [25, 26] intercomparison activity, which involves producing, storing, and analyzing data from multiple climate simulations. It includes data generation, storage, and processing stages.

Application description The overall system is comprised of three stages. In the first stage, climate centers run a prescribed set of common experiments that produce 2-10 PB of data. Data can also come from sensors, in which case the center would still post-process the data before it would be published. Centers can then publish their own output data, or send it to another center to publish. The data is generated over a roughly 2-year period (then post-processed and published over a few more months). This stage is characterized by being both distributed-compute and distributed-data intensive.

The second stage is data storage. The Earth System Grid Federation (ESGF) [27] develops and deploys a federated network of gateways and associated data nodes. As models run at each center, the output data is post-processed into packages with common formats and prescribed metadata conventions. Most centers deploy the ESGF data node software stack, and using this, they manage the data from their experiments. The data node software stack scans the data, makes sure it has the right metadata fields, does QA/QC, and builds a set of catalogs of the prepared data. Minimally, the catalog provides HTTP links to data elements, and it can also provide GridFTP endpoints, and/or product services that abstract the dataset in other ways: get a whole file, subset, browse, etc. When the center is happy with the data/catalog in the data node, they publish it to a host/affiliated gateway. This submits the catalog to the gateway. The gateway then shares this catalog with other gateways so that all gateways have a consistent view of all the published data. In general, most data is replicated at several sites. The replication activity is manually requested/initiated by a gateway owner. The properties of this stage are that it is distributed data-intensive (with distributed gateways and data nodes) and that it is dynamic (data appears in the system over time).

The third stage is data analysis. The approximately 20,000 users (as of Dec. 2010) can browse/search a catalog at any gateway and locate data, which might be hosted by a data node affiliated with another gateway. (This can output a ‘wget’ script that can later fetch the data.) They can also authenticate, and thus gain access to a group, which might have private data. They can also download data via http, GridFTP, or access product services (the latter of which uses ESGF’s data retrieval syntax—DRS—and which can be scripted.) Some users will analyze data from a single model. However, many applications are multi-model analyses, where many users want to look at the same parts of the output of some/all of the models, to understand if and how the models differ. Some centers will gather some/all of the core archive (plus more, perhaps) on local systems for local users to perform “power analyses.” Each user’s data analyses are almost always done on a single system. Because there is no distributed computing infrastructure, this stage is search, access, and transfer intensive.

Big data aspects The overall data generated and stored in the ESGF is 2-10 PB. This includes a core archive, which is 1-2 PB in size and contains the most popular data.

Distributed aspects The data is generated by a distributed set of climate centers. It is stored in a distributed set of federated archives. And it is used by a distributed set of users, who either run data analyses on a climate center with which they are associated, or they gather data from the ESGF to a local system for their analyses.

Dynamic aspects Data is generated over time, so the data in the ESGF changes. One might imagine a future version of ESGF where the launching of data analysis jobs is automated, in which case the location of those jobs would be dynamic, and the system might also be able to respond (or optimize for) various types of applications.

Other important issues This application/infrastructure serves a large group of users, and involves a large amount of political and technical consensus to work.

2.8. *Interactive Exploration of Environmental Data*

In the environmental sciences, the use of visualization techniques is vital for understanding the ever-increasing volume and diversity of data that is being produced by Earth observing systems and computer simulations. The primary purpose of visualization is to gain insight but the majority of scientific tools generate static plots that neither the originating scientist nor the recipient can easily customize to reveal new information. The issues with visualization of a single dataset are compounded if multiple datasets are to be examined simultaneously, as is common in environmental science for model validation, "ground truthing," quality control, and data assimilation.

New, interactive modes of environmental data exploration and visualization based on the principles of simplicity, open standards and user-friendliness are required at all stages of scientific investigation. The application described here is an *infrastructural application* that uses stored data and consists of one stage: data processing. It represents a number of similar applications [28, 29].

Application description The applications often take the form of graphical Geographic Information Systems (GIS) tools, but with better support for large and multidimensional scientific data. The general vision is of a map-based interface, onto which different datasets can be overlain. A processing step may be initiated by rubber-banding an area of the map and selecting from a list of algorithms that process data within the selected area, perhaps calculating statistics of the data. In a desktop application, this processing may take place on the user's desktop, but in a web-based application the processing must take place on a server, which may or may not be co-located with the datasets. There is therefore the common problem of moving large amounts of data around in a distributed system, to which may be added concerns of security in cases in which the data in question are not public.

Big data aspects Data from instruments is small (1–100 MB) but model output may be very big (GB–TB). The data is multi-dimensional in nature. Data aggregated over time from instruments may need to be visualized (e.g. as an animation), causing the visualized data size to grow.

Distributed aspects Increasingly, environmental data are distributed in multiple locations. The applications are driven by data served through distributed web services, that may in turn be "fed" instruments or computer simulation generated data. Compute services need to be integrated with these data services in an efficient, easy-to-use, and transparent way, allowing fast data processing. The processing algorithms are generally simple and high overhead scheduling systems can cause a loss of responsiveness. The infrastructure has more in common with the Web (or cloud) than the grid.

Dynamic aspects Dynamic data are sometimes used, but may arise from real-time feeds from instruments or from looking at live results from a model. Data hosted at multiple locations may be frequently updated (often several times a day). The data services are designed to allow server-side subsetting of large datasets, attempting to minimize the amount of unwanted data traveling across networks. These make it hard to achieve scalability through caching.

In a Web environment, servers and proxy servers (which are sometimes beyond the control of the data provider or the data user) may retain caches of data in a strategy to reduce server load and network traffic. For this reason, the HTTP protocol provides mechanisms for defining expiration times for data resources, specifying when a resource ought to be cleared from the cache. In the environmental and geospatial communities the Open Geospatial Consortium specifications are being widely adopted alongside existing protocols such as OPeNDAP [30]. Unfortunately these protocols, although built atop HTTP, do not make it easy for this versioning to be implemented correctly. The OPeNDAP protocol does not have the concept of a dataset version, meaning that clients have no reliable or efficient means to detect that a dataset has changed. The OGC protocols, in general, *do* provide versioning at the level of the service endpoint, but provide no information on when a resource should be considered expired. Clients are therefore forced to poll the server to check for updates.

Other important issues The application aims for near-real-time interaction with the data, i.e the user should not wait more than a few seconds for some kind of response to a request, so performance with low latency is a key challenge. Security is another serious concern: many environmental datasets are held under access control, and different providers often have very different access control policies. In some infrastructures, these different policies are handled via a role-mapping mechanism [31]. Concerns of access control—as well as data volume—also make it difficult for data to be replicated to different geographic locations (and hence different access control regimes). In a distributed environment, it is common for machines to access data on behalf of end users (e.g., a processing service might download its input data from a remote store). Finding a means for the user to delegate his/her authority to a multi-web-service infrastructure is a key current challenge. The MashMyData project [32] is investigating two solutions, based on Grid proxy certificates and OAuth.

2.9. Power Grids

Demand response (DR) optimization [33] in smart power grids deals with ensuring an adequate supply of electricity by targeted curtailment of power load by consumers during periods of peak load to avoid blackouts. One application used for DR optimization is power demand forecasting at coarse and fine temporal and spatial granularities to allow load curtailment operations to be triggered in advance of a demand-supply mismatch situation. This is a *infrastructural application*. It begins after data generation, and includes the data transformation & decisions, storage, and processing stages.

Application description Power usage information is aggregated from smart meters at consumer premises into the utility's data center [34, 35]. Data is collected at approximately 15-min intervals and transmitted to the utility through cellular and wireless networks using proprietary protocols. Complex event pattern matching over the power usage events and other information source such as weather and scheduling data accessed using web services help perform near term load forecasting. Long term power forecasting at the city-scale or at a smaller micro-grid scale (e.g. industry complex, university campus) is done using power usage models built using machine learning algorithms running at the utility's or micro-grid's datacenter using historical usage data [36]. These applications run on private clouds and are composed using DAG, Complex Event Processing and MapReduce structures [37].

Big data aspects Constructing forecast models using machine learning requires access to historical data that can grow to large sizes [38]. Several hundred TB per year of smart meter power usage data can accumulate for a large city with millions of customers. Other data used in predictive modeling, such as event/people schedules and building/facility features, are smaller, on the order of GBs per year. In future, social network data from consumers may also be mined to learn about power usage patterns. These can also be on the order of GBs.

Real-time forecasting uses smaller data—events on the order of 1 KB, but potentially from millions of sources—that are passed to complex event pipelines or predictive models that run continuously. These pipelines can process GBs of data per hour.

Distributed aspects Data from domains such as power systems, building management systems, weather and traffic, and social networks need to be integrated. While the data are generated and/or stored in distributed locations, they are typically aggregated at the utility's datacenter for analysis. Limited processing may happen locally at the smart meters or local micro-grids with the results pushed as events to the utility.

The applications themselves run on private cloud platforms across hundreds of VMs. Communication between the utility and smart meters is bidirectional, allowing control and pricing signals to be sent from the datacenter to the smart meters. Integrated data will also need to be shared with external applications running on, say, mobile platforms or portals, and with regulatory agencies.

Dynamic aspects The data rates for events published by the sensors may change, either passively due to local measurement limitations or actively throttled by the utility to meet the application's accuracy needs/resource constraints [39]. The data from smart meters can be sent synchronously as they are generated or in batch mode once a day to conserve bandwidth. The application processing itself is done elastic resources, scaling up or down to meet latency requirements. Some of the forecasting models can also decide to listen to additional data streams on-demand when higher accuracy is required or special situations are encountered.

Other important issues Semantic information integration is being performed using an ontology composed together from individual domains [40]. Machine learning models using Matlab, Weka and Hadoop are being used. Complex event processing is using the "Siddhi" CEP engine with semantic support added [41]. Stream processing on Eucalyptus using IBM Infosphere Streams is being performed.

2.10. Fusion (ITER)

With ITER [42] scheduled to operate perhaps around 2022, plasma fusion has become one of the highest priority research items for the US DOE. Scalability of first-principles fusion codes is an important part of fusion research needed to obtain predictive capability of plasma performance with first-principles physics fidelity. The goal is to build a realistic full-distribution kinetic code, in realistic ITER geometry, combining the strengths of each of the existing gyrokinetic codes (GEM, GTS, XGC, and GTC), and utilizing the most up-to-date computer science, hardware design, and applied mathematics technologies developed. This *traditional application* is intended to be capable of 1) discovering macroscopic magnetohydrodynamics activity in burning plasmas, 2) understanding Energetic particle effects in fusion reactors, 3) understanding the heating in a fusion reactor, and 4) understanding the interaction of the plasma at the material boundary. This applications just has the data processing stage, along with associated data movement.

Application description The applications that are currently run (XGC1, XGCP, GTC, GTS) run on all of the DOE and TeraGrid leadership class facilities, and require very large resources for the modest problems that are currently being solved. In the next generation (exascale), not only will all of the computing power that will be available at the time be needed to understand ITER, all of the data that will be generated during the simulation also will have to be understood. The community's techniques have been to examine and understand the data in situ, to reduce the total amount of data produced in the simulation.

Big data aspects Today the team has produced O(100) TB per week for a simulation using the ADIOS middleware [43] incorporated in the simulations. Using staging techniques, they believe that they will only produce O(10) PB on an exascale machine.

Distributed aspects The team always runs on all available resources, so if they have computer time across the country, then their data will be distributed, and so will the computation.

Dynamic aspects The primary aspect that is dynamic is that the set of computing resources that are used for a given run are based on those that are available at the time of that run. As part of this, data is streamed from one running simulation to another, and is transformed in-flight.

Other important issues The applications run on a distributed set of leadership-class facilities, using advance reservations to co-schedule the simulations. Each code reads and writes data files, using ADIOS and HDF5. Files output by each code are transformed and transferred to be used as inputs by other codes, linking the codes into a single coupled simulation. Because the overall data generated is so large, it can not all be written to disk for post-run analysis, so in-situ analysis and visualization tools are being developed.

2.11. Industrial Incident Notification and Response

Applications such as large scale crisis management, maritime security, disease and pollution monitoring, and environmental control need to fuse large amounts of heterogeneous data to form virtual information sources that mitigate noise [44, 45, 46, 47]. The area of greater Rotterdam in the Netherlands is a chemical hub and houses more than 15% of the Dutch population, making safety and security a major concern. Some leakage of chemicals into the air is an everyday, unavoidable event.

The environmental protection agency (EVA) maintains a command and control call center staffed by chemical hazard experts to (1) monitor the air quality and sewage, (2) in case of an incident locate the source, (3) identify the threat and (4) provide the public authorities, the emergency response organization, and the public with situation awareness and suggested actions. The information that is shared is often incomplete, uncertain and inconclusive. Knowledge within the EVA as well as from outside experts and industries must be accessed to meaningfully use that information. This subsection describes the *traditional application* that the EVA uses for these tasks. The application starts after data generation, and includes data transformation & decisions, storage, and processing.

Application description For monitoring and incident detection, continuous streams of data are processed and scanned for anomalies from a small number of physical sensors and air analysis data (gas-chromatographs outputs). If an incident or an anomaly is detected, new streams of data become available from the general public and field inspectors and must be processed. Static information on weather and about chemicals are also needed. These may be stored in different places, access may not be granted automatically, and validity and update frequencies may not match expectations. Thus specialized pre-processing is needed.

Different knowledge-based processing models must be used account for uncertainty in the data and sensor behavior, and provide robust fusion (processing) for situation awareness and decision support. Some confidential data must be processed *in situ*. New intermediate and final data that are generated may have different retention characteristics and need to be stored and made accessible for visualization and analysis in different workflows.

Processing in done in semi-real-time or in batch mode processing on a small, private computer infrastructure. Therefore, scalability is lacking and the dynamics of the workflows is kept under strict human control to match the capabilities of a set of static resources.

Big data aspects The data sizes are not very large.

Distributed aspects A number of special data and sensor interface processes will be distributed over various platforms and fixed in place. A low frequency stream of reasonable fidelity data from a small number of physical sensors and an even lower frequency stream of air analysis data (gas-chromatographs outputs) is continuously trickling in. Low fidelity human provided data from local people complaining to the call center, field inspectors calling in with observations and measurements, information from social media, pictures from cell phones and emails sent to a centralized email account are entered manually into a local databases. Static information such as current weather conditions and forecast and information about chemicals are also needed. These data are made available via a distributed shared data space. Some confidential data must be processed remotely and only aggregated results communicated back to other entities. The information processing algorithms are packaged as autonomous software agents that are managed by a multi agent system infrastructure (middleware) and humans (via GUIs). Processing algorithms need to access multiple concurrently and remote data stores.

Dynamic aspects Both the data and the processing are dynamic in nature. Data that cannot be moved due to confidentiality have to be processed remotely by moving applications to them. A process integration framework will use agents that provide autonomous and composable resources to dynamically construct workflows for data driven processing. If an incident happens or an anomaly is detected, it triggers new streams of data about other anomalies that must be processed. Another source of dynamic behavior is when the event is escalated and additional management processes come into play, each requesting and adding their own sort of processing to the mix.

Other important issues Quality of service aspects that are important for real world and real-time applications are the dynamics of scheduling from an application perspective, multi independent level of security support, processing service discovery support, autonomous reconfiguration support and logging-for-traceability. Most of the dynamic data that these applications use has a short validity period, i.e., it must be processed now or we process the next update coming in. The data is uncertain, i.e., it may have dynamically changing signal to noise ratio and although expected at some base frequency, may not be available for some period of time.

2.12. MODIS Data Processing

This *traditional application* allows the coordinated usage of MODIS [48] satellite imagery, and its integration with ground based sensors and models for environmental applications. It consists of data storage and processing stages.

Application description The application [49, 50] is used to handle the processes of downloading data from different sites and integrating them. The integration consists of re-projecting the data from one map projection to another, spatial and temporal re-sampling, and gap filling. The application itself is a loosely-coupled data pipeline that coordinates these processes.

In addition, scientists can submit Matlab scripts for further custom processing on subsets of the data that has been integrated. This includes some summarization processing and plotting routines. The only time constraints are on the debugging cycle for debugging the scientist-supplied reduction codes. These are purely driven by the users' patience.

The application has been implemented in both the Windows Azure cloud environment [51] and a standard HPC environment.

Big data aspects There is approximately 25 TB of data for one global year. The main data is the various MODIS satellite data products, of which 11 are used by this application. These are in approximately 3,000,000 files for one global year. A smaller set of sensor data from the fluxnet data network is used as input to the evapotransporation calculation, along with a set of 20-30 files containing various metatdata like climate regions used by the calculation.

Distributed aspects The data is collected from a wide variety of FTP servers managed by different groups and brought into the cloud/HPC environment. Custom software tooling manages and validates this download process. Reliable cloud message queues are used to manage the coordination between the different components in the multi-stage pipeline. Simple NoSQL tabular storage is used for logging and fault recovery. Data is stored in cloud blobs and NoSQL table storage. Application scripts on the user's desktop can be uploaded into the cloud as post-processing routines and analyses on the integrated data.

Dynamic aspects The data is currently static and the application acquires the data from a set of FTP servers on a schedule or trigger. But as the pipeline is expanded to handle other data, the usage of real-time streaming sensor data is also expected. There is infrastructure dynamism through the use of cloud virtual machines, acquired and released on-demand, to execute tasks in the pipeline and to execute the user's application scripts. While the pipeline structure itself is static, there is a limited form of application dynamism through the use of user specified scripts pushed into the cloud for execution over the integrated data.

Other important issues The data is in one of two map projections, and multiple temporal and spatial resolutions. The fluxnet sensor data that is used required significant data integration, but was handled in a separate project.

The quality constraints are driven by the needs of the scientific reduction application. Significant effort is required to ensure that the data handled to the scientific application meets these requirements.

2.13. Distributed Network Intrusion Detection

This *traditional application* performs detection of distributed network intrusion and distributed denial of service attacks [52]. Network TCP/UDP traffic data is analyzed/mined at each site. Potential alerts are collected centrally and further analysis is done to see if an attack may be underway by correlating alerts, and events are then communicated back to local sites. The following features are extracted: 1) basic features such as source/destination IP address pair, port number, and protocols and 2) some additional features such as the number of flows from the same source to specific destination IP addresses for a predetermined time period. The extracted features are input for the MINDS system. While well-known intrusions are detected by the known attack detection module, the anomaly detection module is applied to the remaining network connections. During the process of anomaly detection, a training set is generated if not given as input. The anomaly detection module scores each network connection to reflect the level of anomalousness of the network connection compared to the normal network traffic. The highly scored network connections are analyzed by the association pattern analysis module to produce summary and characterization of the possible attacks. This summary and characterization is used to create new signatures and models for unknown emerging attacks. MINDS contains various modules for collecting and analyzing massive amounts of network traffic. Typical analyses include behavioral anomaly detection, summarization, scan detection and profiling. The application includes all four stages: data generation, transformation & decisions, storage, and processing.

Application description MINDS is deployed at each site as a grid service. The analysis application continuously scans local logs, and reports interesting events to a centralized analyzer, and alerts are reported back to local sites.

Each site uses a data capturing device such as network flow tools or network monitoring tools like tcpdump to collect the network traffic data from routers. This collected data is first filtered to remove unimportant network connections due to the large data volume.

All communication in the system is accomplished using messages on top of HTTP.

Big data aspects For local data, this is site dependent. It can be MB/hour/site.

Distributed aspects The system is deployed across clusters located at a set of participating sites (University of Minnesota, University of Florida, and University of Illinois, Chicago) connected by the public Internet. In addition, the data generated is also produced (distributed) at this set of sites.

Dynamic aspects Network traffic data is highly dynamic and constantly streaming in to the application. The data consists of TCP flow and UDP records, which are very structured. They contain IP addresses, ports, various headers, and payloads. Interesting events are propagated across sites. These are very small, a few KB at most.

Other important issues The application is built using Globus Grid Services for wrapping the MINDS analysis applications and for providing storage services for events. At the time the application was written, the project had to develop tools such as remote launching of Grid services themselves. MINDS was also a separate software package that was utilized. The project also built an authorization service for controlling access to data products. Other noteworthy points: real-time is ultimate goal of the project and false positives are more tolerable than false negatives. Finally, the project found it to be very difficult to obtain network data at each site due to privacy concerns.

IDS systems also use pattern detection tools such as Bro and Snort, which perform simple string and regular expression pattern matching on IP packet headers and body. These are monolithic tools that run standalone, without any comprehensive programming support to automate updating the patterns or taking action upon pattern detection.

2.14. Application Summaries

Here we provide short summaries of the computational (as opposed to scientific) aspects of each application in preparation for the analysis that follows in the rest of this document, which is motivated by and centered around these set of applications.

NGS Analytics (§2.1): Data is processed through a customized pipeline of analysis tools, such as BWA, BFAST, Bowtie; processing of each data element is independent of other elements, so processing is decomposed/parallelized over the data elements. All data is stored in files. Later analysis may want to operate on files from multiple data stores distributed around the world, e.g., to compare O(1000s) of TB-scale datasets (metagenomics) with each other. Dynamic elements include the choice of executables in each instance of the pipeline, the design of the pipeline itself, and the choice of computing infrastructure to provide the best resource usage and turnaround time. The solution to this problem is not clear. There is a need for flexible programming systems and associated runtime environments for hardware-software co-optimization.

ATLAS/WLCG (§2.2): There is a hierarchy of systems. Data are centrally stored, and locally cached (and copied to where they likely will be used), perhaps at various levels of the hierarchy. Processing is done by applications that are independent of each other. Processing of one data file is independent of processing of another file, but groups of processing results are collected to obtain statistical outputs about the data.

LSST (§2.3): Data is taken by a telescope. Quick analysis is done at the telescope site for interesting (urgent) events (which may involve comparing new data with previous data). The system can get more data from other observatories if needed, request other observatories to take more data, or call a human. Data (in files) is then transferred to an archive site, which may be at the observatory. At the archive site, the data are analyzed, reduced, and classified, some of which may be farmed out to grid resources. Detailed analysis of new data vs. archived data is performed. Reanalysis of all data is done periodically. Data are stored in files and databases.

SOA Astronomy (§2.4): Services are orchestrated through a pipeline, including a data retrieval service that is used to share data across VO sites, as well as analysis and image processing services. Data (files) are moved through the pipeline, and intermediate and final products can be stored in

AstroGrid storage service, such as mySpace. Scratch space is provided for storage of intermediate results.

Cosmic Microwave Background (§2.5): This application builds a sky map. It reads data from detectors, and obtains data from simulations that are performed on-the-fly, when requested by the map-making component (in a master-worker sense), one of a set of components that gradually reduces the data from the detected and simulated data to a small number of cosmological parameters.

Sensor Network Application (§2.6): Data are collected from mobile sensors that periodically transmit when they come within communication distance of receivers. Fault tolerance (collecting data in a hostile environment) is an important component. The data, which are very diverse, because they can involved different resolutions, sampling frequencies and other management characteristics., are brought to a central site. Stored data are analyzed using statistical techniques, then visualized with tools such as Google Earth.

Climate (§2.7): Climate simulations are run by climate centers. Data (outputs of climate simulations) are distributed in multiple federated stores. Current data analysis and visualization tools require the user to bring the data they want to analyze to a local system, then run the tools locally. New tools will include the capability to add data transfer. This will allow the set of tools to iterate: transfer data, do analysis, etc.

Interactive Exploration of Environmental Data (§2.8): Applications are driven from map-based graphical tools. The inputs are geographic areas and algorithms to run on the data in those areas. Data can come through web services from stored data, sensors, or simulations (possibly running). The services can perform server-side processing to reduce the size of data files to be transferred. Responses are provided in near-real-time. Security also supported through a delegation-based model.

Power Grids (§2.9): Diverse streams arrive at a central utility private cloud over cellular and wireless networks at dynamic rates controlled by the application. Communication is bi-directional, with pricing and control signals sent to customers power meters. A real-time event detection pipeline, involving complex event processing and machine learning/data analysis algorithms and using other stored information such as weather data, can trigger load curtailment operations. Data mining is performed on current and historical data for forecasting. Partial application execution on remote micro-grid sites is possible.

Fusion (§2.10): Multiple computational intensive physics simulation codes run concurrently on a distributed set of parallel computers. Data from some codes are streamed to other codes to link them into a single simulation. Data are transformed in flight into needed inputs. Data are also examined in situ to understand the overall state of the simulation. Some data are stored for later data mining and visualization.

Industrial Incident Notification and Response (§2.11): Data are regularly streamed from diverse sources, and sometimes manually entered into the system. Initial disaster detection causes additional information sources to be requested from that region and one or more applications to be composed based on available data. Some applications run on remote sites (where the data is stored) for data privacy. Escalation can cause more humans in the loop and additional operations.

MODIS Data Processing (§2.12): Data are brought into system from various FTP servers. A pipeline of initial standardized processing steps on data is done on clouds or HPC resources. Scientists can then submit executables and Matlab scripts that do further custom processing on subsets of the data, which likely include some summarization processing (building graphs). The application has been run on a cloud and in an HPC environment.

Distributed Network Intrusion Detection (§2.13): Data are analyzed or mined by a service running at a number of local sites. The results are sent (via HTTP messages) to a central site for analysis. Events are then communicated (via HTTP messages) back to local sites.

3. UNDERSTANDING DISTRIBUTED DYNAMIC DATA

In the previous section we described thirteen applications and analyzed them along an analytical space spanned by the three "D"s of data-intensive, distributed, and dynamic behavior.

Distribution refers to the presence of application data in different physical or logical locations. Distributed data may arise naturally in that the data sources themselves are geographically separated, such as from remotely located sensors. In some cases, data must remain distributed for privacy and policy reasons, or due to constraints in sharing. In other cases, a full dataset may not fit in one location either on disk or in memory and must be decomposed to enable storage and/or computation. Distributed data may be driven by a need for performance, scalability localization, reliability, or availability considerations. For example, copies of data may be generated and distributed to maintain data availability or load balancing. In general, a *distributed application* is one that needs, or would benefit from the use of, multiple resource [53]. Example benefits include increased throughput, decreased time to solution, and increased reliability. A distributed application rarely consist of a single component that is distributed; most often a distributed application is composed of smaller and functionally self-contained, if not entirely independent, components that can execute on different distributed resources.

Dynamism refers to changes in the behavior and characteristics of an entity over time and/or space. Over a long enough timescale, almost all infrastructure, and especially distributed infrastructure, will exhibit some form of dynamism, as a consequence of failure, varying load, etc. However, we are interested in dynamism on timescales that influence application behavior, scheduling decisions, etc. Some prominent examples are resource fluctuation, data availability, and change in application execution characteristics.

In this section we will try to extract commonalities in the dynamic behavior of applications studied in the previous section, as well as properties and types of distribution seen in these applications.

3.1. Types of Distribution

There are many scenarios under which distribution of data is either necessary or desired. In addition to the reasons behind distribution presented at the beginning of this section (viz., large volumes of data) and reasons that constrain or localize data (viz., privacy/policy issues), another common situation under which data is distributed occurs when data comes from multiple sources, for example multiple sensor streams or data centers, but needs to be processed collectively.

There are many ways to address distributed data. These include but are not limited to:

- **Replication:** Data is duplicated on multiple nodes. When fully replicated, each execution unit has access to the same data. Replication can be partial, i.e., data can be replicated to a subset of resources. The degree of replication quantifies the average of the number of copies of each data item. Replication may be non-uniform, e.g., if only a subset of data (perhaps "hot" data) is replicated. Caching is a lightweight form of replication.
- **Partitioning:** Data is divided and distributed across multiple nodes. Different forms of partitioning exist. For example, data can be thematically partitioned with respect to its type, such as sensor data vs. simulated data, or data can be partitioned according to a spatial or temporal attributes. This can be an opaque (e.g., ID) and/or a domain dependent (e.g., spatial) attribute.
- **Streaming**: A form of distributed data processing in which data arrives from a source; often the stream is a steady, continuous flow, but often it fluctuates both spatially and temporally. Also, typically, the stream of data must be processed in near real-time.

Which of the above approaches is employed cannot be reduced to a simple rule. Which option can or should be used often depends upon a plethora of co-dependent factors, such as the data volumes, the *degree of distribution* (which describes the number of data sources and sites used for storing and processing the data), the availability of compute resources, or the desired level of reliability. Data distribution can be difficult to manage for reasons of performance: latencies and bandwidths may vary unpredictably. Further, issues associated with reliability may also need to considered. An example is the occurrence of a partial failure, where some (but not all) of the required data remains available, leads to applications having to decide whether to continue with reduced data or exit without processing what remains available.

Viewed from an application's perspective, there are at least two application characteristics that influence the distribution of data. The first is the stage and the second is the degree of coupling between components of a distributed application. Distribution in the first stage is primarily determined by the structure and layout of the data sources. For example, the placement of the data generated in the climate application (§2.7) is based on which site was assigned to run the particular model that generates the data. But analysis of the data from multiple models requires bringing the data together to a single site. Although it is difficult to formalize the difference due to the large fluctuations between different applications, but in general for a given application, there are differences between the distribution prior to the application's processing stage and the distribution of data within the processing stage, e.g., to aid data-parallel processing.

The second application factor that determines distribution of data is the type of separation and coupling of the components of the distributed application. Different coupling types place different constraints on data distribution, either due to latency, or due to data volumes that must be coordinated. *Loosely-coupled* distribution involves coordination and cooperation between largely independent application components, as might be found in a workflow system. *Tightly-coupled* applications involve greater constraints between the components, either in latency tolerance, coordination requirements or relative scheduling. *Decoupled* distribution results in the separation of application components elements of the same task, for example, sharing a spatial dataset between components that cooperate to perform a single logical task [53]. Most of the applications in §2 are loosely-coupled, with the CMB (§2.5) and Fusion (§2.10) applications as examples of tight coupling.

Table I. Distributed Processing

Application	**Distributed Characteristics**
Next Generation Sequencing (NGS) Analytics	Data sources (NGS machine storage) and compute not always co-located; multiple data sources in case of metagenomics
ATLAS (an example of use of the WLCG)	Source data is generated by one central data source. Data needs to be distributed to match distribution of computing resources. During processing data is copied from closest replica to the resource where compute job is run.
LSST	Multiple different kinds of data sources (observatories, data storage); multi-step workflow that processes and archives data, carried on distributed compute resources.
SOA Astronomy Applications	Data resides in multiple data centers; services are placed closed to the data.
Cosmic Microwave Background	Replication of subset of data to compute resources for processing.
Sensor Network Application	Data is moved from different data sources to a central analysis system for processing
Climate (ESGF)	A subset of the data is commonly moved to compute (desktop or grid resource)
Interactive Exploration of Environmental Data	Data is brought to grid resource or desktop
PowerGrids	Data is brought to central private cloud for analysis
Fusion (ITER)	Simulations run wherever resources are available; in-situ analysis of large output data; partitioning/replication of data for data-parallel processing
Industrial Incident Notification and Response	Distributed data sources. Processing takes place on multiple distributed resources.
MODIS Data Processing	Data is copied (replicated) from different sources for processing. Processing can take place in different locations, e. g. cloud and user desktop.
Distributed Network Intrusion Detection	Distributed data and compute resources.

3.2. *Types of Dynamism*

Table II. Understanding Dynamic Characteristics. This table highlights different types and scenarios for dynamic data in the application set investigated

Application	**Dynamic Characteristics**
Next Generation Sequencing (NGS) Analytics	Data Dynamism: different data formats and sources need to be processed
ATLAS (an Example of Use of the WLCG)	Data Dynamism: new raw data is steadily collected
LSST	Infrastructural Dynamism: Decision making based on observed data
SOA Astronomy Applications	Application and Data Dynamism: data is continuously updated, dynamic queries
Cosmic Microwave Background Mapping	Application Dynamism: dynamic utilization of resources
Sensor Network Application	Data Dynamism: rate and volumes of data dynamic
Climate (ESGF)	Data &Infrastructural Dynamism: new data is generated over time. Dynamic distributed infrastructure for storing and processing of data
Interactive Exploration of Environmental Data	Infrastructural and Data Dynamism: Various data sources, varying input rates and different queries.
PowerGrids	Application, Infrastructural and Data Dynamism: data itself changes, data rates may change (throttling), data sources change (traffic, weather, social networking), data queries respond to seen data (e.g. accuracy adjustment)
Fusion (ITER)	Application and Infrastructural Dynamism: data streamed between simulations (multiple distinct data flows)
Industrial Incident Notification and Response	Application and Data Dynamism: data itself and data rate changes, dynamic data sources, processing changes with data
MODIS Data Processing	Application and Data Dynamism: data that is operated upon changes, infrastructure config changes
Distributed Network Intrusion Detection	Infrastructural, Application and Data Dynamism: data highly dynamically changing, data rate variable, application responds to data, on-the-flight processing

There are often spatio-temporal variability in the values and volumes [54] of data. The applications in §2 have variability in the production/generation rate of data, and in the configuration of data production/generation. Data dynamism may arise from extrinsic factors, for example because sensors begin to observe more events in the same data stream. Contrariwise, data dynamism may occur because an application component requested an increased sampling frequency. Similarly application dynamism may occur through managed replication of a compute-intensive component or from the demands of a particularly complex query. And infrastructure dynamism may arise because the infrastructure expands to meet additional requirements, or contracts because of device failure. A structured analysis of the applications in §2 suggests that the sources of dynamism can be classified into three categories.

1. *Data* dynamism arises when some property of the input data or its delivery changes, for example in terms of arrival rate, provenance, burstiness, or source. There may be variability in the structure of the data, for example, data schema, file formats, ontologies, etc.

2. *Application* dynamism involves changing the processes or components applied to a dataset, and might encapsulate changes in workflow, *ad hoc* queries, or parameterization. There may be variability in the data queries enacted by the application, or in the behavior and/or characteristics of the application.

3. *Infrastructure* dynamism occurs when the system wants to, or is forced to, change its demands on its underlying platform, for example through elastic computation or partial failure. Changes in the structure and load of the infrastructure can be associated with data. Examples include variability in the producer/provenance of the data, in the performance, and in the quality of service of the infrastructure.

Based upon this classification, in Table II, we describe the applications discussed in §2. Many of these dynamic aspects are related: an increase in the rate of data generation may lead an application to increase its degree of parallelism to maintain throughput, which may in turn trigger the allocation of new compute nodes. (In the other, less attractive direction, decreasing computational capability might cause a decrease in parallelism and a consequent need to drop data. Avoiding these cases means treating dynamism equally in both directions.)

This should not be construed to imply that every data-intensive application has all elements of data and compute dynamism and is always distributed, but is a reflection that at extreme scales, dynamism and distribution become increasingly important issues that are often correlated. As alluded to, a high degree of distribution often results in a high amount of infrastructure dynamism, caused by, for example, resource fluctuations and a higher failure probability. Similarly, high levels of dynamism, for example, in the data sources, data rates, and queries, usually correlate with a high degree of distribution. For example, the Interactive Exploration of Environmental Data application (§2.8) manages various sources of data, from archived datasets to real-time feeds. Depending on the spatiotemporal properties of data, computation can be carried out on a set of distributed grid resources or the user's desktop, i.e., distributed processing is used to efficiently handle dynamic data.

3.3. Distributed Dynamic Data Observations

Reviewing Tables I and II, which have summarized the distributed and dynamic properties of the D3 applications, we observe:

- Data generation and consumption are commonly distributed from each other if the volume permits. The Fusion application is a counterexample, due to data volumes and velocity.
- Details of the storage system are often not known. Often, data is stored in a distributed way, e.g., on a distributed file system in the same data center (locally distributed).
- Aggregation of data can either be the equivalent of gather or reduce (using the language of collective communication). Gather is in many ways the reverse process of distribution (called broadcast in the language of collective communication). Similar to distribution, aggregation is a commonly occurring pattern for data analysis.

4. INFRASTRUCTURE

We define *infrastructure* as the hardware and software that is provided to support an application, rather than being explicitly created or hosted by the user themselves. In many cases the precise boundary between infrastructure and application is blurred, e.g., a catalog is a component with functionality that may be similar to that of similar components in other applications, but is often implemented specifically by an application rather than being provided as part of the infrastructure. This may also be the case when reconfiguring an infrastructure service for an application, when it is more onerous than reimplementing the service specifically for that application. We also introduce the concept of an "infrastructure" application as a collection of infrastructure components that are packaged so that they can be used by many applications in defined ways. The difference between these and "traditional" applications is often delineated by who owns and administers the application, with traditional applications "owned" entirely by the user, and infrastructure applications owned by someone other than the user but provided for use by others. Infrastructure can be specialist as opposed to generic, for instance the purpose-built infrastructure in ESGF (§2.7) that is provided to support the climate community.

In this section, we focus on the *software* infrastructure that is provided to support D3 applications, specifically components that deal with distributed or dynamic data. The components that make up the infrastructure can be categorized based on the function provided, namely (i) infrastructure that supports *data management*, (ii) infrastructure that supports *data analysis*, and (iii) infrastructure that supports the structure of the applications. Each of these components may be present at three different levels of the infrastructure: programming frameworks, services and tools (e.g., MapReduce, workflow engines); middleware and platform services (e.g., databases, message queues); and system-level software capabilities (e.g., notifications, file system consistency).

4.1. *Infrastructure to Support Data Management*

Data Sources are used in the acquisition stage as inputs of data into the application. Broadly speaking these sources include types such as files, servers, sensors, instruments, simulations and models. Additionally, databases are often considered as an intermediate data source in some systems. The dynamic characteristics are primarily due to availability. These are often accessed in the Data Generation stage of the application (Figure 1).

Data Storage components are used to provide persistent access to data by the application. They may be structured (e.g., databases, No-SQL tables) or unstructured (e.g., collections of audio recordings). The dynamic characteristics are primarily due to the level of persistence, e.g., temporary, durable, permanent (archived or published), or replicated; and also in the structure of the data stored, e.g., schema free tables as opposed to relational tables. There may also be dynamic characteristics of storage due to rate of growth of data, e.g., a field survey may bring in a lot of data during summer but none during other times. Some storage components like files and databases may also serve as data sources. These are commonly used in the Data Storage stage of the application (Figure 1), but may also be present at the boundaries between the stages for transient staging of data.

Data Access components are gateways to Data Sources and Data Storage components, and provide interfaces that often enforce security controls to restrict access, provide dynamic information about the available data sources and/or data analysis components, and may also perform logging of requests and data movement (e.g., tracking provenance for auditing). A side effect of auditing/logging may be the growth of provenance metadata simply due to access transactions even if no new data is created. Data Access components usually serve at the interface between the application stages, before and after data movement.

Data Movement components are used for bulk or continuous movement of data between parts of the application, or to and from the application, and visible to the application (e.g., staging), or implicit (e.g., automatic replication of files). Components may operate on streams or blocks (files). Dynamic characteristics include partitioning, scheduling and planning (e.g., edge caching of "hot" data, where datasets that are being used frequently during some period are replicated close to the application). Data movement can be considered to be real-time (i.e., as it happens, e.g., sensor data), 'as real-time' (e.g., a storage buffer has been introduced into the transport system, but the data source is still real-time, as with time-series data), or asynchronous (where the ordering of the data movement is independent of the data itself). Latency may be considered as an important characteristic of the data movement infrastructure depending on the application, e.g., real-time processing of sensor data for disaster analysis. Other dynamic aspects of data movement include data rates, their variability from data sources, and the "freshness" of data. These infrastructure appear on the edges between the application stages.

Data Discovery components assist the access to and use of data sources, and include Catalogs, Information Services, and Metadata stores. The dynamic characteristics pertain to the evolution or recalibration of information held in these components. These components are orthogonal to the applications stages and can be used by any of the stages to guide their selection or identification of incoming data sources and outgoing data sinks or storage.

Notification components respond to changes in the application or data and trigger other functionality. These include publish-subscribe message brokers, and complex event processing. By nature, these components support the dynamic aspects of an application. These components again are orthogonal to the applications stages and can be used for coordination of the stages and data movement.

In each case, these components can be single or centralized, distributed, partitioned or federated. These are also typically offered as middleware and platform services, or system-level software capabilities.

4.2. *Infrastructure to Support Data Analysis*

Analysis components include functionality such as comparison, statistical analysis, event analysis (i.e., analyzing information derived from data rather than raw data) and visualization. These components leverage the computational aspects of the infrastructure. Based on the application stages outlined in Figure 1, we identify a number of infrastructure services for data analysis that can be associated with these stages.

Conversion components (associated with the Data Transformation and Decisions stage) deal with the changing of data into a different form, including file conversion, (re-)formatting, transformation and metadata extraction. Heterogeneity of formats and representations is common in many scientific disciplines and such components often provide essential pre-processing of data prior to analysis. These are often referred to as adaptor services (or "shim" services in the Taverna workflow system [55]).

Enrichment components take data and attach additional detail to them, which might include filtering, image processing, information integration, data tagging, semantic annotation, data augmentation, or data fusion (e.g., OLAP). Many of these components may deal with distributed sources of data. It is useful to note that these operations may not be fully automated and may involve a human expert. These components would be associated with the Data Transformation and Decisions stage.

Analytics and visualization components help extract domain knowledge and assist with exploration of the data. These are often closely supervised by the end-user and may even be interactive. The proliferation of "big data" has caused data mining libraries, machine-learning toolboxes and visualization environments to become part of the infrastructure, some of which are deployed for specific domains but others which are more generic (e.g., Weka, GraphLab, Apache Mahout). These components are associated with the Data Processing stage.

4.3. *Infrastructure to Support Coordination within Applications*

Programming abstractions and orchestration frameworks cut across both data management and analysis functions. While different in character from the other infrastructure discussed in this section, they enable the coordination of the different stages of the application, sometimes enforcing a particular constraint to improve, e.g., scalability. These approaches can allow the composition of sequential data processing pipelines and further offer the flexibility of modeling more complex constructs. Standardization of APIs and components within these frameworks can allow interchangeability of components with similar functionality, such as those described in §4.1 and §4.2.

Orchestration components handle the management and choreography of data placement, code placement, communication, and messaging to allow complex analyses to be performed on distributed data sources. These are often composed as workflows or Directed Acyclic Graphs (DAGs) that capture the execution dependency between components. They may also be dynamic if they have the capability to adapt their behavior based on earlier parts of the workflow or can be steered based on external notification received during execution. In addition to supporting choreography, while often making use of one or more resource manager/scheduler components, an orchestration component may also include specialist support for conversion and analysis components discussed above.

Bulk Data-parallel frameworks use "Bulk" operators to express computations over collections of data which are executed in a loosely-coupled, data-parallel manner. The *MapReduce* paradigm [56] and its variants [57, 58] are the most prominent among these, with those like Pregel and Apache

Giraph [59] supporting specific domains like graph analytics. MapReduce forces programmers to model computation in two stages: a *map* phase in which a transformation is applied to every element of a collection independently of every other; and a *reduce* phase in which the transformed results are aggregated through a binary function that is associative and commutative. These properties serve to guarantee that the MapReduce computation can be efficiently parallelized, by performing the map phase concurrently and then reducing without the reduction order affecting the final result.

Stream processing (sometimes called event processing) is an approach modeled on signal processing, in which a stream of data is transformed "on-the-fly" to produce one or more further streams of values. As such it is a programming abstraction suitable for continuous data feeds, or when results need to be pushed to users of interactive applications. Since the data is not assumed to be available *en masse*, it can be used in situations where storing the entire dataset is undesirable, unnecessary, or impossible.

Sensor networks and other intelligent front-end data collection systems are sometimes seen as an extension of stream processing and application pipelines. They often have the feature of being *adaptive*, in that they can vary their exact behavior according either to external control stimuli or from observations made of the data they are themselves observing. Adaptation may also be through users interactively changing the the pipeline, either by changing parameters or by adding or removing tasks in the pipeline. This allows decision logic to be initiated by the pipeline itself, or using an external control signal that may come from users or an instrument. This form of "computational steering" is especially useful in systems being used to explore a dataset. In these cases, it is important to note that the infrastructure is adaptive, rather than the scenario itself.

4.4. Examples of Specific Cyberinfrastructure Usage

We conclude this section by revisiting the application scenarios of §2 in terms of the infrastructure and coordination approaches that they use currently, and discuss elements that could usefully be deployed.

The software infrastructure components we have classified above address different characteristics of D3 applications. Table III summarizes the role of these components in addressing dynamic and distributed characteristics of the D3 applications in §2. Each application makes use of different components, though it can be seen that applications that utilize stream or event processing have larger requirements on notification and orchestration components.

Table IV summarizes the coordination approaches currently dominant in each application case study. It is worth noting that, in common with most long-lived systems, many of the cases have accreted substantial code bases over time and so show little in the way of systematic architecture. (One reason for the dominance of workflow pipelines may be that such architectures are quite suitable for adding extra processing steps to existing computations.)

The applications in Table IV, which make use of scripted or *ad hoc* pipelines, could in most cases be easily replaced with a structured workflow engine. Doing so would bring immediate benefits in terms of reproducibility, but also in terms of standardization of components and interfaces.

A further implication of standardization is that component reuse is greatly improved. This is perhaps best illustrated by observing the increased ability to automate the processing step over repeated runs. The use of standard component-based software techniques such as web services, coupled with web technologies for metadata representation, assist in reuse, re-purposing and integration, for instance into different workflow systems.

The *MapReduce* paradigm [56] and its variants [57, 58] are the most prominent among the bulk data-parallel frameworks which use bulk operators to express computations over collections of data in an loosely-coupled manner. Applications such as Next Generation Sequencing (NGS) (§2.1) and Power Grids (§2.9) use MapReduce for their analytics [61, 62, 38].

There are many examples of stream [63, 64, 65] and event [66, 41, 67] processing frameworks, and recent works have combined streaming with more traditional file-based workflows [68, 69]. Applications such as the Marine Sensor Network (§2.6), Distributed Network Intrusion Detection

Table III. Infrastructure used to support dynamic and distributed properties of application scenarios

Application Name	
Infrastructure to support Dynamic Properties	Infrastructure to support Distributed Properties
Next Generation Sequencing (NGS) Analytics (§2.1)	
Dynamic workload decomposition and distribution for resource optimization *(Orchestration)*	Distributing data from source (generation) to the computing destination (analysis) *(Data Discovery, Movement)*
ATLAS (an example of use of the WLCG) Reconstruction app (§2.2)	
Manage gradual growth of data using SRM *(Data Storage, Access)*	Management, subset replication, data movement and partitioning using gLite and SRM *(Data Storage, Access, Movement, Discovery)*
LSST (§2.3)	
Steering observations, Databases to manage data growth *(Notification, Data Storage)*	Coordinate classification, distributed databases, data movement to distributed observatories *(Notification, Data Access, Data Movement, Data Storage)*
SOA Astronomy Applications (§2.4)	
Catalog contents updated, Query mix changes *(Data Discovery)*	Distributed VO Services for data storage and access *(Data Discovery, Data Access, Data Movement)*
Cosmic Microwave Background (§2.5)	
Dynamic resource selection *(Data Discovery)*	Compute on distributed HPC centers *(Data Movement)*
Sensor Network Application (§2.6)	
Notification of sensor data when in range, data rates and volumes *(Data Movement, Notification)*	Aggregating distributed sensor data *(Data Movement, Notification)*
Climate (ESGF) (§2.7)	
Manage data growth	Distributed, replicated catalogs using ESGF software stack (GridFTP, wget/HTTP) *(Data Access, Data Discovery, Data Movement, Data Conversion)*
Interactive Exploration of Environmental Data (§2.8)	
On-demand data subsetting, caching and versioning, real-time instrument data *(Data Discovery, Data Movement, Data Sources)*	Distributed Data Services, Access Restrictions *(Data Movement, Data Access)*
PowerGrids (§2.9)	
Stream processing over variable sources, rates and volumes, Event processing over different patterns, Trigger analysis on-demand *(Data Sources, Notification, Enrichment, Orchestration)*	Dispersed event sources and sinks, Launch ensemble modeling on distributed clusters and cloud *(Data Sources, Notification, Orchestration)*
Fusion (ITER) (§2.10)	
Stream and transform data between simulation sites on-demand *(Data Movement, Enrichment)*	Run analysis on available, distributed resources *(Data Movement)*
Industrial Incident Notification and Response (§2.11)	
Acquire different data sources and rates, Processing determined by data *(Data Sources, Data Access, Notification, Orchestration)*	Distributed data sources, Analysis by agents across institutions, Access control *(Data Sources, Data Access)*
MODIS Data Processing (§2.12)	
–	Data from distributed FTP sources, Cloud queues for coordination, Partitioned NoSQL data *(Data movement, Data Storage, Orchestration)*
Distributed Network Intrusion Detection (§2.13)	
Analyze dynamic network traffic rates *(Data Movement)*	Run analysis at different sites, Distribute generated data *(Data Movement)*

(§2.13), Power Grids (§2.9), and Industrial Incident Notification and Response (§2.11) use stream and event processing abstractions.

The view of scientific applications as pipelines hides significant differences between (and within) domains, since the different stages of the pipeline receive different emphases in different applications. Some applications must collect and store all the incoming data, with only minor pre-processing to clean the data ahead of the main processing. For others, a major constraint is to reduce the incoming data stream as quickly as possible, perhaps discarding a large fraction of the data before any processing occurs. In spite of these differences, we believe the use of the pipeline abstraction is still useful. Taking some of the examples from §2, medical imaging and astrophysics

Table IV. Coordination approaches used by the applications

Application	Current approaches
NGS Analytics	pipelines that ship with sequencers [60]; workflows [55]
ATLAS	custom algorithms; job submission tied to WLCG
LSST	custom layer; scripted workflows; queries
SOA Astronomy	web services [20]
Cosmic Microwave Background	*ad hoc*
Sensor Network Application	focused on data collection
Climate	*ad hoc*
Interactive Exploration of Environmental Data	visualization tools [29]
Power Grids	stream processing [39]; data mining [38]
Fusion (ITER)	scripted pipeline [43]
Industrial Incident Notification and Response	agent-based [47]
MODIS Data Processing	scripted pipeline [50]
Distributed Network Intrusion Detection	monolithic packages

often fall into the first category, while particle physics (especially in the case of the Large Hadron Collider) often falls dramatically into the second, to the extent that principled data reduction forms one of the major scientific challenges for the project.

There are also significant variations in where "the pipeline" begins and ends. This represents the difficulty in differentiating between "traditional" applications and "infrastructure" applications. Many systems are driven by datasets collected and shared widely, against which several different applications may be applied. Here the infrastructure application is the collection of components used to allow the datasets to be accessed and analyzed by the different applications. Climate science is a good example of this approach. In this case the data is "pre-collected" in the sense of existing outside most of the applications, although its collection may have been a complex task. The raw data required for gene sequencing (§2.1), for example, while being collected using a dedicated and very sophisticated computerized processing system, would generally be regarded as "given" and not part of the processing pipeline, since the pipeline would not affect the collection task. Another way to look at this is that data is collected *independently* of its further processing.

Alternatively the application may be driven directly from a live data stream, with processing happening on the fly. The front-end data collection subsystem is becoming increasingly flexible and capable, with the introduction of sensor networks to replace more passive approaches, and it may often make sense to include the sensor network into the application workflow pipeline, such as for Power Grids and Industrial Incident Response. All these choices have implications for design and provisioning. For instance, the choice of which sort of pipeline a particular application uses may well be fixed, or will change only slowly as the experimental environment evolves. For example, we see some systems that add simulation alongside experimental data collection to act as a verification and validation step.

Revisiting the overall issue of infrastructure, which we defined as the hardware and software needed to run the applications, we note that the specific hardware is not a primary issue. The applications we have discussed all use a mix of the systems that exist today, which have aspects of web, grid, and cloud, and can be considered a hybrid of the three. The key infrastructure components are really the software and services on top of the hardware. In this section, we have discussed infrastructure to support data management and data analysis, as well as the tools and systems used to program applications, and the specific components used for the application in §2. However, there is not much commonality in the software and services that the applications use, so the software and programming infrastructure we have described here is much more broad than what is used by the individual applications we have studied.

5. DISCUSSION AND CONCLUSIONS

This section brings together some key observations from the analysis of applications and infrastructure carried out in this paper. Section 5.1 provides a discussion of dynamic and distributed characteristics of data associated with a set of applications.

We suggest two metrics that may be associated with such applications, primarily as a basis to compare how compute and data processing (in terms of their number and distribution) may be used to find common features between them. Such feature analysis is useful, we believe, for two main reasons: (i) to identify similarities between the applications covered in §2 with others that are in the design phase or under development, so that common requirements can be identified; and (ii) to identify gaps in current software tools and programming systems that such applications use, thereby leading to the development of new tools and systems.

As our application and infrastructure coverage is, by necessity, limited, identifying common features also enables the lessons learned to be generalized and applied more widely. We subsequently discuss a set of architectures for D3 science, and pose some questions we believe are pertinent as the community begins considering the next-generation architectures for D3 applications. In §5.2 we conclude by providing a number of observations to inform applications and cyberinfrastructure developers, based on common trends found in our study.

5.1. Discussion

We summarize four key themes from previous sections: (i) the source and nature of dynamic data observed in the applications surveyed in §2; (ii) the types of data distribution observed in such applications; (iii) the general characteristics that can be associated with D3 science applications; (iv) the characteristics of a computational infrastructure most suited for D3 science applications. Whereas (i) and (ii) are observations that can be made on existing applications discussed in §2, (iii) and (iv) attempt to synthesize general trends that can be applied to applications we have not considered.

5.1.1. Understanding Dynamic Data The applications in §2 show that the dynamic characteristics of many applications arise in two ways: either due to data generation and placement, or due to the certain aspects of the application. The first of these refers to data dynamism and the second to application dynamism (outlined in §3.2).

Data generation, placement and consumption: In many applications, data is dynamically generated in response to a user request or when a particular phenomenon of interest has been observed. Often this data is large in size and must be placed close to the analysis. In some instances, it may not be known where (and when) an analysis request may be made, and it is therefore necessary to identify how replicas of the data could be produced and placed across multiple possible locations. In general, dynamic data arises due to changing data properties, changing data volume (generation, consumption, or processing), and changing data location or distribution.

Application dynamism: In many scientific applications the workflow structure is well defined, but either the components used within this structure or the location of the data that feeds the workflow may change dynamically. Data coordination and planning (identifying how data is managed from ingest through analysis to output) within such applications can range from statically defined interactions between execution units and data sources to a fully adaptable interaction. As outlined in the Industrial Incident Notification and Response application in (§2.11), the workflow structure may also change based on a user request or the event being observed. Data relating to the coordination of a components that make up the application can therefore also be dynamic and change. Exposing the data planning used within an application makes it possible to modify or adapt it over time.

5.1.2. Understanding Distributed Data From §3.1 and Figure 2, we find that applications with distributed data vary in the amount or volume of data distributed, the degree of distribution (scattered over one site or many sites), and the dynamic aspects associated with distribution, e.g., latency and relative scheduling flexibility (and thus tolerance or constraints on data transfer/aggregation).

Some applications use specialized data cyberinfrastructure, while some run utilize existing HPC environments. On the other end of the spectrum are highly dynamic applications that process sensor data and make near-realtime decisions based on streaming data, e.g., the power grid and sensor network applications.

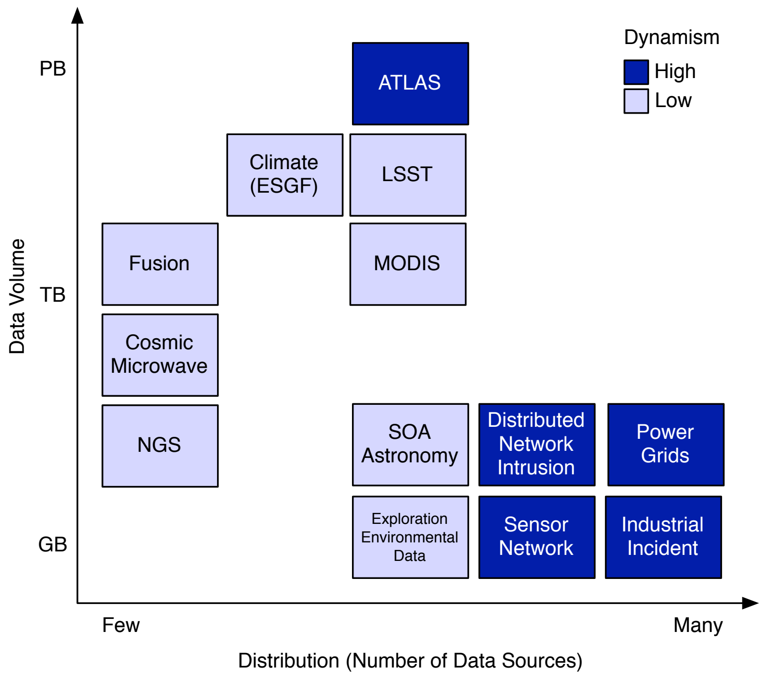

Figure 2. The distribution, dynamism, data volume characteristics of the surveyed D3 applications. The clustering of applications with high dynamism in the bottom right hand corner indicates a similar structure amongst these applications, namely multiple (sensor-based) sources of data generation with temporal variation in data generation. Although we do not explicitly color/shade traditional and infrastructural applications, it is worth mentioning that there is no correlation between application type and any of the three "D"s; we attribute this to a strong dependence of degree-of-distribution infrastructure available for both types of applications.

Data topology can be used to identify the flow of data through the application, with nodes representing execution units and edges representing communication features. One aspect of managing data topology is "data planning," which refers to how the data is coordinated from ingest through analysis to output. In all cases, there is a coordination of the data with code used to analyze or transform it. In some cases these form clear, integrated stages where data flows from one to another. In others the way that data enters and flows through the application is more loosely defined and the execution of code is independent. Some examples include the ATLAS experiments, which have a single point of primary data generation which fans out to multiple analysis sites followed by mass exchange of derived data, and sensor grids, which have multiple sources of data generation collected to a single point of analysis followed by constrained point of reuse.

Some applications must collect and store all incoming data, with only minor pre-processing to clean the data ahead of the main processing. For others, a major constraint is to reduce the incoming data stream as quickly as possible, perhaps discarding a large fraction of the data before any processing occurs.

Although dynamic and distributed properties have been characterized separately, it is important to remember, as the above patterns indicate, that distribution and dynamism become increasingly correlated at extreme scales. For example, applications that run on distributed infrastructures are often also highly dynamic and vice versa.

5.1.3. Understanding Distributed Dynamic Applications In §3 we characterized dynamic and distributed properties of the D3 Science applications. Our analysis was qualitative by necessity, due to the fact that approaches to quantify dynamism and distributed properties were often intimately tied to application/scenario definition and specific usage modes. Notwithstanding, to quantify better dynamic properties of an application, we believe a term analogous to the use of Amdahl's number [70], such as [Number of Compute Operation / Amount Data (Consumed, Generated or Transformed)] could be used to characterize 'static' data-intensive applications and systems. Note that this term is scale-invariant, i.e., a value of 1 could correspond to very small or very large values of both numerator and denominator, and that it does not capture the time scales over which data changes.

Similarly, another term [Amount of Data Transferred / Degree of Distribution] could be used to quantify the "scale" of distribution, namely the degree of distribution as a function of the amount of data transferred, where the degree of distribution can be defined as the number of distinct data locations.

It has proven difficult to provide "global" quantitative values of these ratios, as often the value of the numerators is subject to how an application/scenario is defined, even more so for a "class of applications" as opposed to a specific application. We believe, nonetheless, that these measures suggest a possible way to quantify an application's dynamism and distribution and thereby find possible common approaches solutions. We plan to explore this in our future work, and also hope others will consider building on these initial concepts.

5.1.4. Towards an Architecture for D3 Science? An investigation of data-intensive distributed applications on production cyberinfrastructure, leads us to believe that there are three primary macroscopic architectures and approaches for distributed data-intensive applications: (i) localize all data into a "cloud" (or a large analytical engine) – the paradigm adopted by several genome projects and the cancer atlas, (ii) decompose and distribute data to an appropriate number of computing/analytical engines as available – the paradigm employed by particle physics for the discovery of the Higgs Boson (similar to the application in §2.2), and (iii) a hybrid of the above two paradigms, wherein data is decomposed and committed to several infrastructure, which in turn could be a combination of either of the first two paradigms, similar to what is done in the Power grids application (§2.9).

It is obvious that the first architecture can be used for large volumes of data, but there are self-evident limitations to the scalability or validity of this model. How does this limitation change, qualitatively or quantitatively, if dynamic data and/or applications are considered? This begets the question: how fundamental an issue is *dynamic*?

The high-energy physics (HEP) community has self-organized towards the second architectural approach. Although distributed computing infrastructures representing this approach, such as OSG and EGI, have been successful for this community and application, broad uptake across other application types, modes, and data volumes has not taken place. What lessons does the HEP experience teach us for the other communities, such as bioinformatics? Can an infrastructure conforming to a particular architecture and developed for a specific application domain be generalized to another science domain, using the same underlying macro architecture but by providing a different set of services and specialized data cyberinfrastructure? Or is the architecture specific to a *fixed* range of compute-data characteristics? If so, what are these characteristics?

5.2. Conclusions

The growth in data volumes and its importance is having profound implications on the way applications are designed and the way we develop infrastructure and provision associated services.

Many fields, such as biology, that until recently were not characterized by their intensive use of data are rapidly becoming data-driven.

Associated with the growth in data volumes are increasing levels of dynamism and distribution; sometimes distribution is logical, but sometimes the data is physically distributed for reasons of performance or convenience. At large scales, dynamism becomes intrinsic to the application and infrastructure; sometimes it occurs because of changing experimental or analytic conditions, sometimes due to the availability of infrastructure and resources.

An important motivation for this work is to capture the state of the art in the design and development of distributed dynamic applications and the associated infrastructure currently used for their execution. We have aimed to try to discern common approaches and challenges, and to identify any obvious gaps (or opportunities). Using a common vocabulary and terminology to describe otherwise distinct and unrelated applications allows potential application users/developers to benefit from the insights that a common analytical framework provides. For example, seeing how applications with similar patterns (e.g., pipelines) have addressed issues of dynamism and distribution enables common solutions and facilitates the emergence of best practices even in the absence of formal methods. Similarly, our analysis sends a useful message to domain and application scientists that the challenges and barriers they face are not unique even if current solutions require a level of customization. In this paper, we have identified many different applications characteristics, programming systems, and infrastructure techniques that either support dynamic data or produce it.

In §2, we divided applications into groups: traditional and infrastructural applications. Traditional applications have been built by one or more authors in order to solve a particular problem or to answer a science question. Infrastructural applications have been built by groups or communities to allow the solution of a set of problems, or to answer a set of science questions. All applications in general, but infrastructural applications in particular cannot be developed in vacuum; they must be executed on shared infrastructure, and are thus sensitive to the external tools and services, as well as their provisioning as software-systems.

As part of this survey, we observe that there exist patterns in the described applications; however, implementation and deployment specific details can often mask these patterns. For example, applications often differ in the way they handle distribution and dynamism, such as managing such as data/compute localities. This makes it difficult to support patterns of distribution and dynamism in a general-purpose fashion, for anything but the simplest patterns.

ACKNOWLEDGEMENTS

The work was sponsored by NSF OCI-1059635 and made possible by the UK e-Science Institute Research Theme on Dynamic Distributed Data-Intensive Programming Abstractions and Systems. SJ contribution was also funded by NSF CAREER ACI-1253644. We thank Simon Dobson, Jon Weissman and Jon Blower for useful initial discussions, as well as 3DPAS workshop attendees. Discussions of the selected applications benefited from the generous help of Adam Barker, Jon Blower, Julian Borrill, Silvia Delgado Olabarriaga, Steve Fisher, Keith Jackson, Scott Klasky, Bob Mann, Don Middleton, Kees Nieuwenhuis, Manish Parashar, Stephen Pascoe, and Jon Weissman, though any errors are the responsibility of the authors. Some of the work by Katz was supported by the National Science Foundation while working at the Foundation; any opinion, finding, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

1. John D. McPherson. Next-generation gap. *Nat Meth*, 6(11s):S2–S5, 11 2009.
2. Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, and Durbin R. The sequence alignment/Map format and SAMtools, 2009. http://www.htslib.org.

3. R.C.G. Holland, T. Down, M. Pocock, A. Prlic, D. Huen, K. James, S. Foisy, A. Drger, A. Yates, M. Heuer, and M.J. Schreiber. BioJava: an open-source framework for bioinformatics. *Bioinformatics*, 2008. http://biojava.org.
4. BioPerl. http://www.bioperl.org.
5. .NET Bio. https://github.com/dotnetbio/bio.
6. Tom Oinn, Mark Greenwood, Matthew Addis, M. Nedim Alpdemir, Justin Ferris, Kevin Glover, Carole Goble, Antoon Goderis, Duncan Hull, Darren Marvin, Peter Li, Phillip Lord, Matthew R. Pocock, Martin Senger, Robert Stevens, Anil Wipat, and Chris Wroe. Taverna: lessons in creating a workflow environment for the life sciences: Research articles. *Concurr. Comput.: Pract. Exper.*, 18(10):1067–1100, August 2006.
7. Ilkay Altintas. Distributed workflow-driven analysis of large-scale biological data using biokepler. In *Proceedings of the 2nd international workshop on Petascal data analytics: challenges and opportunities*, PDAC '11, pages 41–42, New York, NY, USA, 2011. ACM.
8. Belinda Giardine, Cathy Riemer, Ross C. Hardison, Richard Burhans, Laura Elnitski, Prachi Shah, Yi Zhang, Daniel Blankenberg, Istvan Albert, James Taylor, Webb Miller, W. James Kent, and Anton Nekrutenko. Galaxy: A platform for interactive large-scale genome analysis. *Genome Research*, 15(10):1451–1455, October 2005.
9. John Ellis. Beyond the standard model with the LHC. *Nature*, pages 297–301, 19 July 2007.
10. ATLAS Collaboration and G. Aad et al. The ATLAS experiment at the CERN large hadron collider. *Journal of Instrumentation*, 3(08):S08003, 2008.
11. Massimo Lamanna. The LHC computing grid project at CERN. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 534(1-2):1–6, 2004. Proceedings of the IXth International Workshop on Advanced Computing and Analysis Techniques in Physics Research.
12. Z. Ivezic, J. A. Tyson, E. Acosta, R. Allsman, S. F. Anderson, J. Andrew, R. Angel, T. Axelrod, J. D. Barr, A. C. Becker, J. Becla, C. Beldica, R. D. Blandford, J. S. Bloom, K. Borne, W. N. Brandt, M. E. Brown, J. S. Bullock, D. L. Burke, S. Chandrasekharan, S. Chesley, C. F. Claver, A. Connolly, K. H. Cook, A. Cooray, K. R. Covey, C. Cribbs, R. Cutri, G. Daues, F. Delgado, H. Ferguson, E. Gawiser, J. C. Geary, P. Gee, M. Geha, R. R. Gibson, D. K. Gilmore, W. J. Gressler, C. Hogan, M. E. Huffer, S. H. Jacoby, B. Jain, J. G. Jernigan, R. L. Jones, M. Juric, S. M. Kahn, J. S. Kalirai, J. P. Kantor, R. Kessler, D. Kirkby, L. Knox, V. L. Krabbendam, S. Krughoff, S. Kulkarni, R. Lambert, D. Levine, M. Liang, K. Lim, R. H. Lupton, P. Marshall, S. Marshall, M. May, M. Miller, D. J. Mills, D. G. Monet, D. R. Neill, M. Nordby, P. O'Connor, J. Oliver, S. S. Olivier, K. Olsen, R. E. Owen, J. R. Peterson, C. E. Petry, F. Pierfederici, S. Pietrowicz, R. Pike, P. A. Pinto, R. Plante, V. Radeka, A. Rasmussen, S. T. Ridgway, W. Rosing, A. Saha, T. L. Schalk, R. H. Schindler, D. P. Schneider, G. Schumacher, J. Sebag, L. G. Seppala, I. Shipsey, N. Silvestri, J. A. Smith, R. C. Smith, M. A. Strauss, C. W. Stubbs, D. Sweeney, A. Szalay, J. J. Thaler, D. Vanden Berk, L. Walkowicz, M. Warner, B. Willman, D. Wittman, S. C. Wolff, W. M. Wood-Vasey, P. Yoachim, H. Zhan, and for the LSST Collaboration. LSST: from Science Drivers to Reference Design and Anticipated Data Products. *ArXiv e-prints*, May 2008.
13. Intermediate Palomar Transient Facility, http://www.ptf.caltech.edu/iptf.
14. N. M. Law, S. R. Kulkarni, R. G. Dekany, E. O. Ofek, R. M. Quimby, P. E. Nugent, J. Surace, C. C. Grillmair, J. S. Bloom, M. M. Kasliwal, L. Bildsten, T. Brown, S. B. Cenko, D. Ciardi, E. Croner, S. G. Djorgovski, J. van Eyken, A. V. Filippenko, D. B. Fox, A. Gal-Yam, D. Hale, N. Hamam, G. Helou, J. Henning, D. A. Howell, J. Jacobsen, R. Laher, S. Mattingly, D. McKenna, A. Pickles, D. Poznanski, G. Rahmer, A. Rau, W. Rosing, M. Shara, R. Smith, D. Starr, M. Sullivan, V. Velur, R. Walters, and J. Zolkower. The Palomar Transient Factory: System Overview, Performance, and First Results. *PASP*, 121:1395–1408, December 2009.
15. A. Rau, S. R. Kulkarni, N. M. Law, J. S. Bloom, D. Ciardi, G. S. Djorgovski, D. B. Fox, A. Gal-Yam, C. C. Grillmair, M. M. Kasliwal, P. E. Nugent, E. O. Ofek, R. M. Quimby, W. T. Reach, M. Shara, L. Bildsten, S. B. Cenko, A. J. Drake, A. V. Filippenko, D. J. Helfand, G. Helou, D. A. Howell, D. Poznanski, and M. Sullivan. Exploring the Optical Transient Sky with the Palomar Transient Factory. *PASP*, 121:1334–1351, December 2009.
16. Yogesh Simmhan, Roger S. Barga, Catharine van Ingen, María A. Nieto-Santisteban, Laszlo Dobos, Nolan Li, Michael Shipway, Alexander S. Szalay, Sue Werner, and Jim Heasley. Graywulf: Scalable software architecture for data intensive computing. In *HICSS*, pages 1–10, 2009.
17. David Jewitt. Project Pan-STARRS and the outer solar system. *Earth, Moon and Planets*, 92:465–476, 2003.
18. Yogesh Simmhan, Catharine van Ingen, Alex Szalay, Roger Barga, and Jim Heasley. Building reliable data pipelines for managing community data using scientific workflows. In *International Conference on eScience (eScience)*, pages 321–328. IEEE, 2009.
19. International virtual observatory alliance. http://www.ivoa.net/.
20. Adam Barker. *Flexible Service Composition*. PhD thesis, University of Edinburgh, 2007.
21. A. A. Penzias and R. W. Wilson. A Measurement of Excess Antenna Temperature at 4080 Mc/s. *Astrophysical Journal*, 142:419–421, July 1965.
22. C. M. Cantalupo, J. D. Borrill, A. H. Jaffe, T. S. Kisner, and R. Stompor. MADmap: Fast Parallel Maximum Likelihood CMB Map Making Code. *Astrophysics Source Code Library*, page 10018, October 2011.
23. M. Cronin and B. J. McConnell. SMS seal: A new technique to measure haul-out behaviour in marine vertebrates. *Journal of Experimental Biology*, 362(1):43–48, 2008.
24. A. Lindgren, C. Mascalo, M. E. Lonergan, and B. J. McConnell. Seal-2-Seal: A delay-tolerant protocol for contact logging in wildlife monitoring sensor networks. In *Proceedings of the Fifth IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, 2008.
25. Intergovernmental Panel on Climate Change. *Fourth Assessment Report: Climate Change 2007: Synthesis Report: Summary for Policymakers*. Intergovernmental Panel on Climate Change, 2007.
26. K. E. Taylor, R. J. Stouffer, and G. A. Meehl. An overview of CMIP5 and the experiment design. *Bull. Am. Meteorol. Soc.*, 93:485–498, 2012.
27. Dean N. Williams, Karl E. Taylor, Luca Cinquini, Ben Evans, Michio Kawamiya, Michael Lautenschlager, Bryan N. Lawrence, Don E. Middleton, and the ESGF contributors. The earth system grid federation: Software framework

supporting CMIP5 data analysis and dissemination. *CLIVAR Exchanges*, 56:40–42, 2011.
28. J. D. Blower, A. Gemmell, K. Haines, P. Kirsch, N. Cunningham, A. Fleming, and R. Lowry. Sharing and visualizing environmental data using virtual globes. In *UK e-Science All Hands Meeting 2007*, pages 102–109, September 2007.
29. J. D. Blower, A. Santokhee, A. J. Milsted, and J. G. Frey. Blogmydata: a virtual research environment for collaborative visualization of environmental data. In *UK e-Science All Hands Meeting 2010*, September 2010.
30. OpenDAP. http://www.opendap.org.
31. B. N. Lawrence, R. Cramer, M. Gutierrez, K. Kleese van Dam, S. Kondapalli, S. Latham, R. Lowry, K. O'Neill, and A. Woolf. The NERC DataGrid prototype. In S. J. Cox, editor, *Proceedings of the U.K. e-Science All Hands Meeting*, 2003.
32. MashMyData. http://www.mashmydata.org.
33. David Kathan et al. Assessment of demand response and advanced metering. Technical report, Federal Energy Regulatory Commission, 2012.
34. Yogesh Simmhan, Viktor Prasanna, Saima Aman, Sreedhar Natarajan, Wei Yin, and Qunzhi Zhou. Towards data-driven demand-response optimization in a campus microgrid. In *Workshop On Embedded Sensing Systems For Energy-Efficiency In Buildings (BuildSys)*. ACM, 2011. Demo.
35. Yogesh Simmhan, Saima Aman, Baohua Cao, Mike Giakkoupis, Alok Kumbhare, Qunzhi Zhou, Donald Paul, Carol Fern, Aditya Sharma, and Viktor K. Prasanna. An informatics approach to demand response optimization in smart grids. Technical report, University of Southern California, 2011.
36. Saima Aman, Yogesh Simmhan, and Viktor K. Prasanna. Improving energy use forecast for campus micro-grids using indirect indicators. In *International Workshop on Domain Driven Data Mining (DDDM)*, 2011.
37. Qunzhi Zhou, Yogesh Simmhan, and Viktor Prasanna. Scepter: Semantic complex event processing over end-to-end data flows. Technical report, University of Southern California, 2012.
38. Wei Yin, Yogesh Simmhan, and Viktor Prasanna. Scalable regression tree learning on hadoop using OpenPlanet. In *International Workshop on MapReduce and its Applications (MAPREDUCE)*, 2012.
39. Yogesh Simmhan, Baohua Cao, Michail Giakkoupis, and Viktor K. Prasanna. Adaptive rate stream processing for smart grid applications on clouds. In *International Workshop on Scientific Cloud Computing (ScienceCloud)*, pages 33—38. ACM, 2011.
40. Qunzhi Zhou, S. Natarajan, Y. Simmhan, and V. Prasanna. Semantic information modeling for emerging applications in smart grid. In *Ninth International Conference on Information Technology: New Generations (ITNG)*, pages 775–782, 2012.
41. Sriskandarajah Suhothayan, Kasun Gajasinghe, Isuru Loku Narangoda, Subash Chaturanga, Srinath Perera, and Vishaka Nanayakkara. Siddhi: a second look at complex event processing architectures. In *ACM workshop on Gateway computing environments (GCE)*, 2011.
42. ITER. International thermonuclear experimental reactor (iter), http://www.iter.org.
43. J. Lofstead, S. Klasky, Schwan K., N. Podhorszki, and C. Jin. Flexible IO and integration for scientific codes through the adaptable IO system (ADIOS). In *CLADE 2008 at HPDC*, June 2008.
44. Tina Comes, Michael Hiete, and Frank Schultmann. A spatial scenario-based multi-criteria decision support system for strategic emergency management. In *Proecedings of the 73rd meeting of the european working group multiple criteria decision aiding*, April 2011.
45. Tina Comes, Claudine Conrado, Tiphaine Dalmas, and Niek Wijngaards. Robust scenario-based multi-criteria decision support in strategic emergency management. In *Proceedings of the 8th international conference on information systems for crisis response and management (ISCRAM2011)*, May 2011.
46. Ate Penders, Gregor Pavlin, and Michiel Kamermans. A collaborative approach to construction of large scale distributed reasoning systems. *International Journal on Artificial Intelligence Tools (IJAIT)*, 20:1083–1106, 2011.
47. Gregor Pavlin, Niek Wijngaards, and Kees Nieuwenhuis. Towards a single information space for environmental management through self-configuration of distributed information processing systems. In *Proceedings of the TOWARDS eENVIRONMENT Opportunities of SEIS and SISE: Integrating Environmental Knowledge in Europe*, pages 94–103, 2009.
48. Vincent V. Salomonson, W. L. Barnes, Peter W. Maymon, Harry E. Montgomery, and Harvey Ostrow. MODIS: Advanced facility instrument for studies of the earth as a system. *IEEE Transactions on Geoscience and Remote Sensing*, 27(2):145–153, 1989.
49. Microsoft. MODISAzure. http://research.microsoft.com/en-us/projects/azure/azuremodis.aspx, 2010.
50. Jie Li, M. Humphrey, D. Agarwal, K. Jackson, C. van Ingen, and Youngryel Ryu. eScience in the cloud: A MODIS satellite data reprojection and reduction pipeline in the Windows Azure platform. In *IEEE International Symposium on Parallel and Distributed Processing (IPDPS)*, pages 1–10, April 2010.
51. Microsoft. Windows Azure. https://azure.microsoft.com/.
52. Jon B. Weissman, Vipin Kumar, Varun Chandola, Eric Eilertson, Levent Ertoz, Gyorgy Simon, Seonho Kim, and Jinoh Kim. DDDAS/ITR: A data mining and exploration middleware for grid and distributed computing. In *Workshop on Dynamic Data Driven Application Systems, Proceedings of 7th International Conference Computational Science - ICCS 2007*, pages 1222–1229, 2007.
53. Shantenu Jha, Murray Cole, Daniel S. Katz, Manish Parashar, Omer Rana, and Jon Weissman. Distributed computing practice for large-scale science and engineering applications. *Concurrency and Computation: Practice and Experience*, 25(11):1559–1585, 2013. http://dx.doi.org/10.1002/cpe.2897.
54. Dirk deRoos, Chris Eaton, George Lapis, Paul Zikopoulos, and Tom Deutsch. *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*. McGraw-Hill, 2011.
55. Taverna: open-source and domain-independent workflow management system. http://www.taverna.org.uk/.
56. Jeffrey Dean and Sanjay Ghemawat. MapReduce: simplified data processing on large clusters. In *Proceedings of the 6th USENIX Symposium on Operating System Design and Implementation*, 2004.

57. Jaliya Ekanayake, Hui Li, Bingjing Zhang, Thilina Gunarathne, Seung-Hee Bae, Judy Qiu, and Geoffrey Fox. Twister: a runtime for iterative MapReduce. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*. ACM, 2010.
58. Yuan Luo, Beth Plale, Zhenhua Guo, Wilfred W. Li, Judy Qiu, and Yiming Sun. Hierarchical MapReduce: Towards simplified cross-domain data processing. *Concurrency and Computation: Practice and Experience*, 2012.
59. Grzegorz Malewicz, Matthew H. Austern, Aart J.C Bik, James C. Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. Pregel: a system for large-scale graph processing. In *International Conference on Management of data*, SIGMOD '10. ACM, 2010.
60. Illumina. Genome analyzer data analysis software. http://www.illumina.com/documents/products/datasheets/datasheet_genome_analyzer_software.pdf.
61. Michael C. Schatz. CloudBurst: highly sensitive read mapping with MapReduce. *Bioinformatics*, 25(11):1363–1369, 2009.
62. Aaron McKenna, Matthew Hanna, Eric Banks, Andrey Sivachenko, Kristian Cibulskis, Andrew Kernytsky, Kiran Garimella, David Altshuler, Stacey Gabriel, Mark Daly, , and Mark A. DePristo. The genome analysis toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data. *Genome Res.*, 2010.
63. Sirish Chandrasekaran, Owen Cooper, Amol Deshpande, Michael J. Franklin, Joseph M. Hellerstein, Wei Hong, Sailesh Krishnamurthy, Samuel Madden, Vijayshankar Raman, Frederick Reiss, and Mehul A. Shah. TelegraphCQ: Continuous dataflow processing for an uncertain world. In *CIDR*, 2003.
64. Daniel J. Abadi, Yanif Ahmad, Magdalena Balazinska, Ugur etintemel, Mitch Cherniack, Jeong-Hyon Hwang, Wolfgang Lindner, Anurag Maskey, Alex Rasin, Esther Ryvkina, Nesime Tatbul, Ying Xing, and Stanley B. Zdonik. The design of the Borealis stream processing engine. In *CIDR*, pages 277–289, 2005.
65. Chee S. Liew, Malcolm Atkinson, Jano van Hemert, and Lianxiu Han. Towards optimizing distributed data streaming graphs using parallel streams. In *International Workshop on Data Intensive Distributed Computing (DIDC)*, 2010.
66. EsperTech Inc. Esper - complex event processing. http://www.espertech.com/esper/.
67. Microsoft Corp. Microsoft streaminsight. msdn.microsoft.com/en-us/library/ee362541.aspx, 2010.
68. D. Zinn, Q. Hart, T. McPhillips, B. Ludändscher, Y. Simmhan, M. Giakkoupis, and V.K. Prasanna. Towards reliable, performant workflows for streaming-applications on cloud platforms. In *11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2011)*, 2011.
69. Chathura Herath and Beth Plale. Streamflow programming model for data streaming in scientific workflows. In *IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, 2010.
70. Jim Gray and Prashant Shenoy. Rules of thumb in data engineering. *International Conference on Data Engineering*, 2000.
71. Dynamic distributed data-intensive programming abstractions and systems research theme. http://wiki.esi.ac.uk/3DPAS.
72. Workshop on dynamic distributed data-intensive applications, programming abstractions, and systems (3DAPAS). https://sites.google.com/site/3dapas/.
73. Workshop on D^3 science. http://www.ci.uchicago.edu/D3Science/.

APPENDIX: METHODOLOGY

This paper originated in work at the UK e-Science Institute (eSI) at the University of Edinburgh, in a research theme examining Dynamic Distributed Data-intensive Programming Abstractions and Systems, called 3DPAS [71]. The description of this theme's work was:

> Many problems at the forefront of science, engineering, medicine, and the social sciences, are increasingly complex and interdisciplinary due to the plethora of data sources and computational methods available today. A common feature across many of these problem domains is the amount and diversity of data and computation that must be integrated to yield insights.
>
> For many complex data-intensive applications, moving the data may have restrictions. Increasingly important type of data-intensive applications are data-driven applications. For example, data is increasingly large-scale, distributed arising from sensors, scientific instruments & simulations. Such data-driven applications will involve computational activities triggered as a consequence of independent data creation; thus it is imperative for an application to be able to respond to unplanned changes in data load or content. Understanding how to support dynamic computations is a fundamental, but currently a critical missing element in data-intensive computing.
>
> The 3DPAS theme seeks to understand the landscape of dynamic, distributed, data-intensive computing: the programming models and abstractions, the run-time and middleware services, and the computational infrastructure. It will analyze existing tools and services, identify missing pieces and new abstractions, and propose practical solutions and best practices.

The theme held three workshops, and in each, application scientists and computing technologists were invited to give presentations on their work, and then to be involved in discussions to organize and understand the presented materials. Between the workshops, the theme organizers also drafted outlines of a report that was a predecessor of this paper, for discussion in the next workshop. Overall, there were about 50 workshop attendees.

In order to select applications and gather information about them, during the various workshop discussions, when an application that "felt" new was mentioned, we asked the person who mentioned it to provide a written set of answers to the following questions about the application:

1. What is the purpose of the application?
2. How is the application used to do this?
3. What infrastructure is used? (including compute, data, network, instruments, etc.)
4. What dynamic data is used in the application?
   (a) What are the types of data,
   (b) What is the size of the dataset(s)?
5. How does the application get the data?
6. What are the time (or quality) constraints on the application?
7. How much diverse data integration is involved?
8. How diverse is the data?
9. Please feel free to also talk about the current state of the application, if it exists today, and any specific gaps that you know need to be overcome.

We then used the text of these answers to describe each of the applications (in §2) in terms of:

- What does the application do? What problem does it solve?
- How does it do it? How does the application run? What infrastructure does it need/use?
- What are the big data aspects of the application?
- What are the distributed aspects of the application?
- What are the dynamic aspects of the application?
- What else is important about the application? What does it do well? Poorly? How was the app written? What tools does it use?

After the workshops, the theme organizers and a set of attendees who were interested in participating started discussing how to organize and analyze the applications and the technologies, including infrastructure, programming systems and abstractions. These participants are the current authors of this paper, and this paper is the result of that process.

As part of the theme, two 3DPAS workshops [72, 73] were organized at major conferences: HPDC 2011 and IEEE eScience 2011. These workshops had both peer-reviewed and invited papers.

A combination of invitation-only focused discussion meetings and broader community engagement events were used to ensure that the applications surveyed and ideas captured in this paper are representative of current community thinking as well as of intellectual relevance.