Chapter 4: Data Management Architectures

Terence Critchlow, *Pacific Northwest National Laboratory*
Ghaleb Abdulla, *Lawrence Livermore National Laboratory*
Jacek Becla, *SLAC National Accelerator Laboratory*
Kerstin Kleese-Van Dam, *Pacific Northwest National Laboratory*
Sam Lang, *Argonne National Laboratory*
Deborah L. McGuinness*, Rensselaer Polytechnic Institute*

Data management is the organization of information to support efficient access and analysis. For data intensive computing applications, the speed at which relevant data can be accessed is a limiting factor in terms of the size and complexity of computation that can be performed. Data access speed is impacted by the size of the relevant subset of the data, the complexity of the query used to define it, and the layout of the data relative to the query. As the underlying data sets become increasingly complex, the questions asked of it become more involved as well. For example, geospatial data associated with a city is no longer limited to the map data representing its streets, but now also includes layers identifying utility lines, key points, locations and types of businesses within the city limits, tax information for each land parcel, satellite imagery, and possibly even street-level views. As a result, queries have gone from simple questions, such as "*how long is Main Street*?", to much more complex questions such as "*taking all other factors into consideration, are the property values of houses near parks higher than those under power lines, and if so, by what percentage".* Answering these questions requires a coherent infrastructure, integrating the relevant data into a format optimized for the questions being asked.

Data management is critical to supporting analysis because, for large data sets, reading the entire collection is simply not feasible. Instead, the relevant subset of the data must be efficiently described, identified, and retrieved. As a result, the data management approach taken effectively defines the analysis that can be efficiently performed over the data. To support the variety of complex query specifications required by different types of analysis, specialized data management architectures have evolved over time. These predominantly software-centric architectures enable efficient searching across large data sets for well-defined subsets of the data. This chapter gives an overview of the approaches to large-scale data management currently in popular use.

Section 1 presents an overview of large-scale file systems, including the standard interfaces and libraries that have been designed to support large-scale experimental and simulation data sets, as well as the challenges these systems face. This is the most general data management architecture, but also the hardest to use effectively. It is particularly suitable for problems where either raw throughput is critical, or where data can be easily organized into files that will be read in their entirety. For many applications, significant increases in efficiency can be found by optimizing the layout of data on disk through a preprocessing step. Geospatial information is not only large, but often requires complex analytics such as computing distance and overlap between regions. To address these needs geospatial database systems, discussed in Section 2, have been developed and deployed. These systems typically use specialized data structures hidden from the user to efficiently support the complex queries they perform. No discussion of

large-scale data management technology would be complete without an overview of relational and next-generation database management systems. Relational systems form the basis for most business data management infrastructures, as well as for many scientific environments. These systems use a layer of abstraction, relational tables, to hide the complexity of storing and accessing categorical data. As we discuss in Section 3, however, this basic abstraction is being extended to manage multi-modal data as well as queries beyond the traditional standard SQL select-project-join specifications. Finally, we conclude this chapter with a discussion of the role of metadata in supporting data access. As Section 4 demonstrates, effective use of metadata can determine whether or not a problem is tractable.

Section 1) Data Storage and Architectures

A major part of the data management picture includes storing data persistently and reliably on behalf of the application. Large computing systems include a storage area known as *the storage system* for placing datasets. In data intensive computing, the storage system plays a vital role of not only providing a location for persistent and reliable storage of data, but also the means to organize information that enables efficient access and analysis. The hardware devices and network connections that make up the storage system consist of many storage components, provide the capacity necessary to store massive datasets and enable parallel access to data.

Data intensive computing applications often distribute one large coordinated computation across many compute elements and periodically dispatch I/O accesses from these compute elements in parallel. The underlying data intensive computing architectures vary widely, from high performance computing systems used for computational science and advanced computation to massive enterprise data centers that run complex data mining codes on petabyte datasets. These architectures differ in their capacity, performance and workload requirements, but they all share basic design principles fundamental to supporting and managing massive datasets common to the various types of data intensive applications. To frame our discussion of the storage system, we will focus on storage architectures typically deployed within high-performance computing. Many of the design principles we will describe in the HPC storage system carry over into large enterprise data centers and other systems as well.

Solving problems in science and technology that are too large or too complex to be addressed through laboratory experiments, computational science applications generate massive quantities of data, and have a wide variety of data models and access patterns. A typical I/O workload in HPC might consist of mining a large dataset for desired information, with many queries on the same dataset executing concurrently. Datasets could be anywhere from terabytes to a few petabytes in size and so are too large to fit entirely in memory. The storage system that must support this workload must not only provide enough capacity to store terabytes of data, it must provide enough parallel bandwidth to allow access to these large datasets quickly. It must also provide a degree of reliability, ensuring that data will not be lost or corrupted should failures of various storage system components occur. The challenges of capacity, performance, and reliability are seen across all areas of data intensive computing and as a result, the various storage architectures built to meet these challenges have similar design criteria.

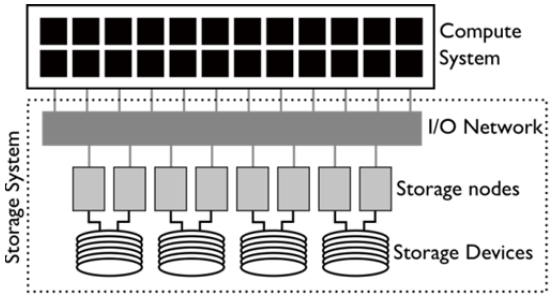

Fig. 1. Typical I/O system in high performance computing

Storage systems in HPC are built from many individual hardware components, enabling parallel access and preventing single points of failure. The storage hardware is often physically partitioned from the compute system, with separate machines dedicated to managing secondary storage for the cluster. These machines, shown as the *storage nodes* in Figure 1, are connected to the compute cluster or front-end nodes through a high-speed I/O network. Each storage node manages a subset of the storage devices, which may consist of directly attached RAID controllers and a few disk drives, or network attached enterprise storage controllers with hundreds of disks. The I/O network, storage nodes, and collection of storage devices make up the storage system of large data centers and HPC clusters, providing persistent data storage for applications. This design enables high performance from all the compute elements in the cluster, and allows the I/O system to scale as needed. Current installations at some of the largest supercomputing centers in the world consist of hundreds of storage nodes and thousands of disk drives [TOP509, LCL09], while some of the largest enterprise data centers manage petabytes of data across thousands of storage nodes [CDG+06]**.**

Computational science applications typically store their datasets in files. As an unstructured stream of bytes, a file can be easily split up by the storage system to optimize performance and provide versatility for a wide variety of I/O workloads. Some applications lay out data so that each compute element stores parts of a dataset in a separate file, or they may store all the application datasets in only a few files.

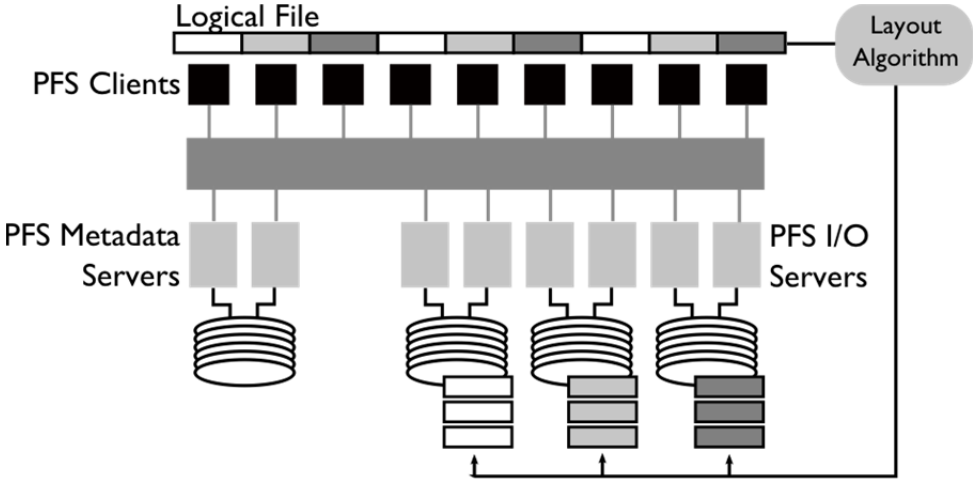

Fig 2. Components of the parallel file system.

A parallel or distributed file system brings together the various components of the I/O hardware to provide applications with a single file interface to secondary storage. It is designed to provide parallel and high-performance access to data, while ensuring a consistent view of the data across all the compute elements in the cluster. Figure 2 illustrates the parts of the parallel file system (PFS) software running on the hardware components of the I/O system. The PFS client software executes on the compute elements, and communicates over the I/O network to the PFS I/O and metadata servers running on the storage nodes. The I/O servers provide direct access to file data from the clients, while the metadata servers maintain file metadata and the directory structure, and organize the layout of file data across the I/O servers. Parallel file systems either manage file data across I/O servers in units of fixed sized bytes (known as blocks) [SH02], or they manage data across I/O servers in objects of variable size, as is the case with Lustre [S03], PanFS [WUA08], and PVFS [LCL09].

Parallel file systems provide direct access to the data at an I/O server through well-defined layout patterns stored at the metadata server. This allows PFS clients to write or read data by direct communication with the I/O servers, avoiding the metadata server as a potential bottleneck. In most cases, the parallel file system uses a simple round-robin layout scheme, so that a file becomes uniformly distributed across I/O servers as the file grows. This allows an application to access different regions of the file from different I/O servers, ensuring that no single I/O server becomes bottlenecked by a multitude of I/O requests from many compute elements. This performance optimization is so ubiquitous in parallel file systems that it has been adopted by the NFS specification [SEN10], a standard file system networking protocol. The specification includes standardized layout schemes (collectively called parallel NFS or pNFS) allowing direct access to I/O servers from file system clients.

Parallel file systems must be resilient to failures. Storage node or storage device failures should not cause data loss or prevent applications from accessing their data. To provide this degree of resilience, parallel file systems rely partly on the storage devices to be fault-tolerant. Large supercomputing centers invest in expensive storage devices with built-in redundancy mechanisms to ensure that a failed disk drive does not cause a loss of data. Multiple paths to the data guard against failures of controllers, and specialized hardware performs rebuilds of failed disk drives. The parallel file system provides high-availability by integrating a failure detection and automatic recovery, enabling it to detect a failed storage component, and provide access to needed data through alternative paths.

The standard POSIX file interfaces [POS96] (e.g. *open()*, *read()* and *write()*) exported by parallel file systems and used by many HPC applications require *sequential consistency* [CSG99] between writes, where each write operation must be atomic and appear to have occurred in the same order by all compute elements. This requirement places a greater burden on the parallel file system than other types of file systems, because a file's data is distributed across the I/O servers and requests are made directly from clients. In order to provide consistent behavior between concurrent requests from two or more clients, many parallel file systems provide exclusive access to regions of a file by granting a lock for those regions to one client at a time. This ensures consistent behavior, but can limit performance as clients must first acquire a lock from a lock manager running on a separate server before performing requests to the I/O servers. While locking regions of the file can hinder performance for some I/O workloads, it does allow the PFS client to cache the locked file regions, which has shown to improve performance for applications that perform many small I/O requests. With a lock acquired, small requests to a file can be

aggregated together so that a single large request can be sent to the I/O servers. Still, file systems that implicitly lock file accesses to provide sequential consistency not only require extra communication to grant lock requests, but must also coordinate between competing compute elements, adding significant complexity to the file system. While sequential consistency is necessary in some distributed applications, for computational science it most often provides stronger guarantees than are usually required. Extensions to the POSIX I/O standard [GWG05] have been proposed to address this issue, known as *Lazy I/O*. Lazy I/O relaxes POSIX sequential consistency for certain I/O accesses, allowing applications to choose the degree of consistency required.

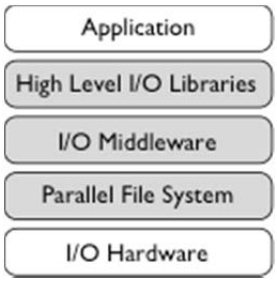

Fig. 3. The I/O software stack in HPC.

Because the I/O workloads of computational science have such intensive and unique data requirements, a simple POSIX interface on-top of the parallel file system is usually not enough to provide high-performance I/O to the application. Instead, a suite of I/O libraries and software are utilized to improve performance and convenience for applications as they perform I/O to storage. Figure 3 shows the HPC I/O software stack. The parallel file system manages the I/O hardware components as we have already described. Above the parallel file system, I/O middleware aggregates and optimizes I/O accesses from the application to the file system. MPI-IO [TGLU99] and ADIOS [LKS+08] are examples of I/O middleware used in HPC. Lastly, the high-level I/O library provides I/O interfaces that fit with the application's data models, allowing it to access data based on multi-dimensional variables instead of as a flat stream of bytes. We will describe each of these software interfaces in more detail here.

HPC applications often coordinate computation and share state between compute elements using the message-passing interface (MPI) [GHL98]. This provides an opportunity for coordinating accesses to files directly instead of requiring the file system to perform that coordination on the application's behalf. The MPI-IO interface [TGLU99] was designed with this goal in mind. By leveraging the existing MPI framework, MPI-IO is able to ensure consistent access across process in the MPI application. MPI-IO is also able to optimize I/O through message passing. Multi-dimensional arrays can be distributed across compute elements with individual processes getting many small regions of the dataset. This leads to many small I/O accesses, which are not optimal for parallel file systems. MPI-IO optimizes these small I/O accesses by designating a few compute elements as buffering nodes so that many small requests can be aggregated into a single large I/O request before sending the request to the file system [TGL99]. In MPI-IO this is known as *collective buffering* because the compute elements act collectively to perform I/O through the buffering nodes.

High-level libraries (HLLs) allow computational science applications to store their datasets in a much more natural way than writing bytes to a file. Instead, applications using HLLs can generate and store

data directly into *variables*, and the library takes care of converting the structured data of a variable into a flat stream of bytes to be stored in a file. This not only simplifies the task of storing large datasets, it also allows the HLL to optimize the layout of the structured datasets in the file for performance. Two common HLLs used in computational science are pNetCDF [LCR03] and HDF5 [HDF5]. Both libraries support parallel I/O access to structure datasets using MPI-IO to aggregate and optimize I/O accesses to the parallel file system.

Computational science applications tend to generate datasets that match the scale at which they run. Larger scales allow applications to examine problems at finer granularities, or solve larger, more complex systems. This means that applications will generate larger and larger datasets as supercomputing systems increase in capability and performance. Datasets generated on the largest systems today are terabytes in size, often stored in millions of files. If trends continue, by 2020 the largest supercomputing systems will consist of 100 million application tasks and have somewhere between 20-50 petabytes of memory [KBB08]. This presents a challenge to the storage system to provide not only the bandwidth required to generate and store these large datasets, but also to place the data within the storage system efficiently so that it can be easily accessed during analysis. Storage systems will need to organize and manage data that improves locality of access as well as continue to provide reliability and performance.

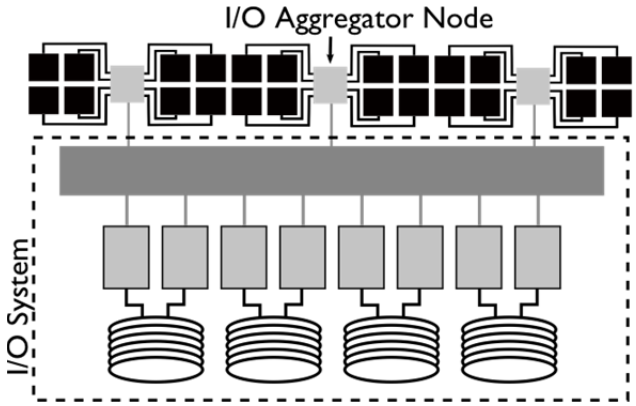

Fig 4. I/O Aggregator nodes.

Many of the challenges arising in HPC storage systems today are performance related. Parallel file systems often struggle with applications that choose to share a dataset in a single file across thousands of compute elements in the system. This type of workload can cause contention for locks to shared data regions and reduce system throughput to a fraction of peak bandwidth [LW]. Storage systems also struggle to support simultaneous parallel access to millions of files, due to the heavy load on the metadata server this workload creates [CLR09]. As the number of individual compute elements increases exponentially over the next decade, these problems will only get worse. To address the increase in core counts, storage systems researchers have added a buffering layer for the file system. Similar to collective buffering in MPI-IO, *I/O aggregation* provides dedicated nodes within the compute system to perform I/O on behalf of a portion of the compute elements. Instead of the parallel file system receiving I/O requests from each of the compute elements, it receives fewer and larger requests from the I/O aggregator nodes. Figure 4 shows the architecture of an HPC system with I/O aggregator nodes. Along with separate hardware components, I/O aggregation requires special I/O forwarding software to efficiently

aggregate and buffer the I/O requests from compute elements. The *I/O forwarding scalability layer* (IOFSL) [ACI09] is a joint project to provide portable and highly efficient I/O forwarding software.

The I/O demands of data intensive computing create other challenges as well. To support capacity and performance requirements, systems are being deployed with tens of thousands of disk drives and thousands of storage nodes. Storage systems will only be able to scale with compute performance by continuing to increase this count of individual components, in turn increasing the likelihood of failure of any single component. Storage systems face a daunting challenge to provide storage that is resilient to failures while still maintaining the highest possible degree of performance.

These challenges will not be met easily. Most data intensive applications quickly consume all I/O and storage resources made available and experts anticipate that they will continue to do so. Only the right mixture of system designs and software will allow storage systems to continue to scale to exabyte datasets and beyond.

Section 2) Spatial and Temporal Information

Spatial data is data about objects in space, which can be points, lines, regions, or volumes. A feature set defines attributes that are associated with spatial objects, for example date, temperature, humidity, altitude, longitude, latitude are features that could be associated with a city. The object's feature values and spatial location can be updated over time, adding a temporal dimension. An instance in the time dimension captures the state of the object at that time and the feature values for an object over multiple time instances represent the history of that object as it evolved over time. The time dimension can quickly cause a data explosion. For example high-resolution environmental sensing and high resolution satellite or telescope imaging can produce terabytes to petabytes of data each day (e.g. the LSST is expected to collect 20 TB/night [LSST]); the amount of data stored increases linearly with the number of days recorded. This in turn introduces several challenges including reliable spatio-temporal data storage, efficient and scalable retrieval algorithms, efficient data analysis algorithms, and data provenance management (including defining, querying, and controlling access to provenance data) [HTT09].

If a spatial object has fixed location overtime then it is referred to as a *static* object; otherwise, it is a *moving* object. We will motivate the spatio-temporal data management challenges using tracking of moving objects as an example. Data sets for moving objects and trajectories for moving objects (e.g. GPS-tracks or celestial objects' tracks) are either collected from devices such as cell phones or GPS's or extracted from spatial images. In the case of GPS or cell phones, the data can be used to create hour-by-hour census for cities to facilitate understanding human behavior in urban areas. In the case of Astrophysics or medical images, behavior modeling is not as important, however, keeping track of objects, their location, and how the attributes' values change is still important. Tracking of moving objects and predicting future locations for these objects can introduce several challenges especially since the behavior of the moving object depends on the application domain. For example, movement of astronomical objects is predictable in most cases (a planet or a comet will follow a certain trajectory). Theoretically, three spatial data points can be used to predict the next location of the spatial object; however, several real-world complexities could make the prediction erroneous. First, simply identifying which objects are the moving objects is challenging [KKA+09] and requires matching objects with high certainty using non-spatial features. Then, if the object passes behind or in front off another spatial object non spatial features need to be used to help resolve the objects identity. For objects such as cars or trucks, predicting the next location is even harder since the object can make sudden turns or stops. Fusing data

with other information sources such as road maps or traffic information could help in tracking a moving object and predict its next location.

Spatio-temporal data poses several interesting research challenges and it is useful to classify the research challenges into categories:

1- Data management algorithms which include algorithms for efficient data storage and retrieval.
2- Spatio-temporal data mining.
3- Query processing and optimization for spatio-temporal data.
4- Hardware and software architectures for spatio-temporal data.
5- Algorithms for real time sensor spatio-temporal data.
6- Ontology and structure of spatial data.
7- Data provenance.
8- Uncertainty quantification, representation and propagation in spatio-temporal databases.
9- Formal models and languages for representing spatio-temporal data.

Although there is a lot of work that spans these 9 areas, there is little work that has been done with respect to large data sets, with one exception - scalable data mining algorithms. For space considerations we consolidate these 9 research areas into the following three general challenges and present the current state of the art in these areas in relation to anticipated exascale data sets.

1. Performance related challenges, which include categories 1, 3, 4
2. Spatio-temporal data analytics and real time analysis; which includes categories 2 and 5.
3. Uncertainty quantification representation and propagation in spatio-temporal databases.

After discussing the ongoing research in these areas, we conclude this section with an example of spatio-temporal analysis based on work done for the Large Synoptic Survey Telescope project. This example illustrates how these approaches may be applied in a real-world setting.

Performance related research and challenges

This area of research includes, but is not limited to, benchmarking of spatio-temporal databases, query processing and optimization, and efficient data storage and indexing. After observing that existing spatio-temporal benchmarks lack the ability to thoroughly evaluate the temporal capabilities and assume that temporal events are evenly spaced in time, [PW98] proposes a new set of spatio-temporal queries that require significant temporal processing and evaluate the ability of a database to handle three dimensional data. In addition to evaluating the database performance, the proposed 36 queries can evaluate the impact of the operating system enhancements including file systems, virtual memory management, and process scheduling enhancements on the query performance. [CJL08] notes that success in designing efficient database-indexes relies on empirical studies and proposes a benchmark for evaluating the indexing of current and near-future positions of moving objects. While benchmarking measures the performance of the data management systems, the underlying spatio-temporal access methods implemented in these systems is are major factor in affecting the performance.

[MGA03] is a short, yet excellent, survey of spatio-temporal access methods and it provides a classification of these methods that distinguishes between access methods that index the past (static data) and ones that indexes current and future predictions (streaming data). Static access methods deal with time in three different ways, first, as another dimension where the focus is on handling spatial queries efficiently while temporal queries are considered less important. The RT-tree, which is based on the R-

tree for spatial indexing and the TSB-tree for temporal indexing are examples of this class of access methods [XHL90].

Second, the temporal dimension is dealt with as a separate dimension and each time instance has its own spatial tree. The MR-tree, which is an optimized implementation using overlapping B-trees of a series of R-trees over time implements this data structure.

Third, access methods are focused on objects that move over time and the data structure is optimized for trajectory queries. For indexing current position, several access methods have been proposed, however, the differences between these methods are primarily due to scalability of these algorithms. Most of these methods, however, try to separate the historical data from the current data. An example of such access methods is the LUR-tree. For indexing current and predicting future positions, extra information, such as velocity and destination, needs to be stored. In some cases, the object's movement can be modeled by a linear equation, while in other cases it could have multiple future locations (e.g. car coming to an intersection) that needs to be updated in real time once more information is available. The PMR-quadtree is an example of an access method that can be used to index future trajectories. The survey concludes that more work should be done to support operators that are important to the spatio-temporal domain such as nearest-neighbor queries and spatio-temporal joins.

Most of the work in industry focuses on building specialized distributed systems that will scale traditional DBMS's. For example, Oracle TimesTen [OT] is an in-memory database that uses a traditional database architecture, and its main advantage is that it relies on large memory to cache the data that will be processed. Netezza [NET], on the other hand, took a different direction by using "active disks" to speed up database operations. Netezza is distinguished by employing FPGAs and PowerPC processors rather than standard disk controller chips and Intel CPUs to build their own customized hardware. A Netezza system distributes data across the disks and queries it in parallel, at the disk level, using the FPGAs.

In summary, to meet the exascale data analysis challenge, more spatio-temporal benchmarks and use case studies are needed to explore and uncover the weaknesses in the current database architectures. Specifically, proposed benchmarks should be able to uncover weaknesses of a data management system with respect to the specific data intensive application requirements. For example if the application will need to answer a set of different spatio temporal queries with both space and time constraints, the time to rebuild the index or redistribute the data must be considered in the benchmark. More use cases are clearly needed in the distributed data processing area; for example, there is a need for efficient and dynamic declustering algorithms to improve parallel data analysis performance. Similarly, data preparation (loading or redistributing), which can be time consuming and expensive, has a significant impact on query performance. Finally, R-tree indexing is implemented in most of the modern database systems; more complex indexing schemes may be implemented as customized stored procedures within the database system but at the cost of performance. The trade-off between making these algorithms (such as the zones algorithm [GSS]) native to the database versus building customized solutions must be thoroughly evaluated.

Spatio-temporal data analytics research and challenges

Spatial data mining is the application of data mining techniques to spatial data with the objective of finding patterns in spatial data. Spatio-temporal data mining is similar but with the goal of finding patterns across both spatial and temporal domains. The work in this area is motivated by the need to identify complex, time-sensitive patterns hidden in large data sets in domains as varied as sensor streams, climate data, astrophysics, and traffic monitoring. Currently, the work in this area handles spatio-temporal data the same as any other type of data without special consideration for the semantics of the

spatial and temporal dimensions. This approach is attractive since the clustering and other data analytic algorithms can be used to immediately explore the data. However, the semantics of the spatio-temporal relationships are lost. Ideally, a theoretical framework can be developed to capture the relationships of the spatio-temporal components to the other data, providing a powerful analysis capability.

Spatio-temporal data mining is a very active field and there are an increasing number of publications in the popular data mining conferences such as SIGKDD. [RHS07] developed a bibliography of the publications in the spatio-temporal data mining field divided into nine categories which reflects the wealth of research in this area. Among the identified categories are time series mining, association rule mining in spatio-temporal data, and discovery of temporal patterns

There are several new domains of science that will benefit from spatial-temporal data mining in particular. For example, [TPRB03] discusses an interesting, but small scale clinical application of spatial data mining: a classification tree is used to classify a sample of normal patients versus patients with corneal distortion caused by keratoconus. Spatial features that can help classify normal patients versus patients with corneal distortion are extracted from images of patients' eyes. The shape of the cornea is modeled using Zernike polynomials and the coefficients of the polynomial are used as features for a decision tree classifier.

[NKKMP08] provides an overview of spatio-temporal data mining research. It uses highway-traffic to introduce spatio-temporal queries and concepts that need to be supported by a spatio-temporal database. The authors distinguish between local pattern mining, where the objective is to search and find interesting spatio-temporal patterns, for example a periodic pattern for a group of people travelling together from one city to another, and global pattern mining where the objective is to mine the full data for all patterns or to create predictive models. The first task is similar to information retrieval where the objective is to match a specific pattern within a specific context, while the later task uses clustering and classification techniques to extract global patterns across the data (exploratory activity).

To move beyond the current state of the art and enable complex spatio-temporal analysis, a systematic approach that integrates spatio-temporal semantics, for example the notions of proximity and temporal order (e.g. an event comes before or after another event), into data management frameworks is needed. Formally defining these relationships and operation (e.g. intersects, overlaps, contains) will allow data mining algorithms to reason over the relationships between spatial objects and to identify interesting and emerging spatio-temporal patterns.

Uncertainty quantification, representation, visualization, and propagation in spatio-temporal databases

The literature includes several approaches to support uncertainty in spatial data including visualizing uncertainty [MRGMGH05], models and data types for uncertainty in spatial databases [S08], and spatial data mining with uncertainty. What uncertainty means for spatio-temporal data is still unclear, with at least two common definitions being used. In the first, uncertainty is defined in terms of fuzzy spatial objects; i.e., objects that do not have very well defined boundaries such as polluted areas. The location, center, and boundaries of these objects can be represented with range of values or with error bars. In [S08], an abstract data model to handle fuzzy objects introduces fuzzy operations that can be used to manipulate these objects. As a first step, spatial objects can be defined with an uncertainty measure as one of the attributes. For example a fuzzy *point* that belongs to an air-polluted cloud, can have a numeric attribute that measures the degree of pollution of that point. This, in turn, decides the strength of membership of the point to the cloud and helps define the cloud's (or region's) boundaries. Spatial

operations, such as intersects or contains, can also be implemented while considering the uncertainty. A fuzzy spatial *algebra* is proposed that can be used to address the uncertainty in the spatial operations.

Realizing the challenges of visualizing uncertainty in spatial data, [MRGMGH05] reviews and assesses progress toward visual tools and methods to help analysts manage and understand information uncertainty. The paper discusses the value of visualizing uncertainty and raises two related questions: *"Is it helpful to include uncertainty in the visualization?"* and *"Do users with different knowledge level use uncertainty differently?"* For example, do domain expert understand a particular visualization of uncertainty better than a novice user?

Similar to other research areas in the spatio-temporal domain, the area of spatio-temporal data mining with uncertainty is an open research area. More work on the semantics of uncertainty with respect to the spatial objects is needed to fully develop the field. For example, formal models, such as the one suggested in [S08], need to be extended, generally accepted, and ultimately standardized. Before gaining acceptance, these models need to be evaluated against new use cases to test their completeness. In the context of database design, more work is needed on data structures that can support fuzzy manipulation of spatial objects and indexing techniques.

Crossmatching of astronomical objects

In this section we describe our experience evaluating different database architectures to implement the crossmatch algorithm (i.e. identification of the same, moving object across different images taken at different times) for a large data set. The Large Synoptic Survey Telescope (LSST) is an example of a data intensive application that drives towards examining new hardware/software data intensive architectures because it combines two important challenges: large data volumes and real-time or near-real-time data querying and analysis. Furthermore, when fully operational, the multi-petabyte data set captured by the LSST is expected to be used by many research studies from the astronomy community. As a result, efficient solutions to the LSST data capture, processing, cataloging, and analysis pipelines are likely to have broad scientific impact.

Our task is to identify the same object in different images of the sky. In addition to the object moving relative to other objects within the image, these images are taken at different times, possibly days or weeks apart, and under different conditions (e.g. cloud cover, angles). To do this, we start by comparing objects based on their distances from each other. This requires fast retrieval of candidate objects from a large historical catalog. Once the candidate objects have been retrieved, the crossmatch operation tries to match the new object to this historical data. The focus of our evaluation was to determine the best architecture to identify the candidate historical objects.

We examined different database system configurations and indexing strategies to speed up the data access and processing operations. First, we investigated techniques for constructing spatial indices on large streaming data. We looked at the performance of a spatial index library (SaIL [HHT05]) for creating spatial indices on large data volumes as they are ingested, and for efficiently searching and retrieving objects using these indices. We then developed parallel versions of the index creation and querying steps provided with the library distribution. The spatial indexing was customized – by selecting an appropriate value for the fan-out factor of the tree-based index – for optimal performance based on the analysis steps that follow it. While this was a promising direction, we concluded that the performance of the spatial index will be limited by the query selectivity. In our case we have to rebuild the distributed index to get the targeted performance, and unfortunately this is a very expensive operation and cannot be avoided.

Second, we evaluated a set of database systems with different architectural configurations and identified performance differences and bottlenecks in these systems for spatial crossmatch operations. We investigated the execution of two database-oriented crossmatch algorithms, the zones algorithm [GSS] and the optimized zones algorithm [BLMNA07] on three different architectures (see figure 5):

1- Netezza Performance Server R , a parallel database management system with active disk architecture support for certain types of database operations,
2- MySQL Cluster, a database system designed for high-availability and high-throughput, and
3- a distributed collection of independent database system instances with support for data replication. In this configuration, we combined the best features from the two previous configurations to achieve the best performance for this use case. The achieved performance is not specifically unique to the MySql server and it can be obtained using other database engines if used in the same way.

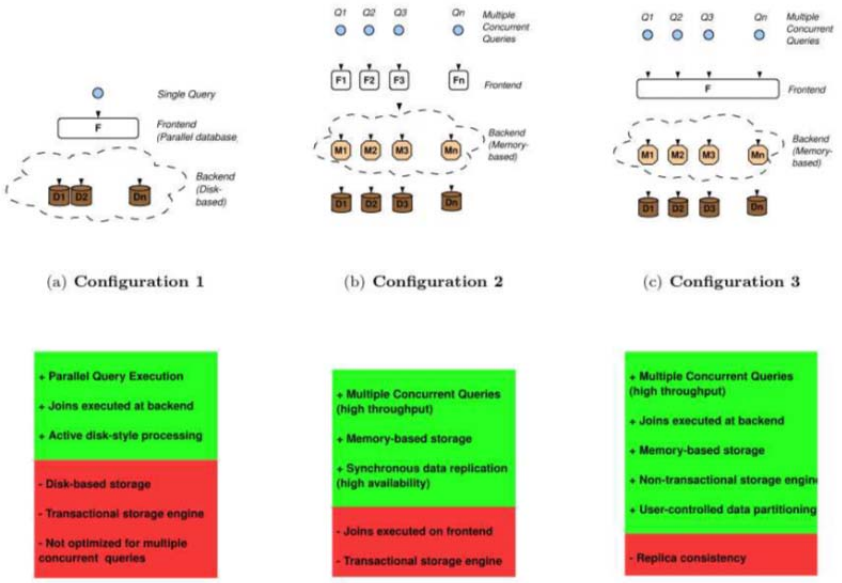

Figure 5: Alternative Database Architecture Configurations, their pros and cons

The tested algorithms are very well know to the astronomy community and were developed by a team from the computer science and the astronomy fields. For test data, we use the USNO-B catalog (a public astronomy catalog generated by US Naval Observatory at Flagstaff containing over billion objects) since it closely emulates the LSST data when it becomes operational [KKA+09].Our experimental evaluations were based on real queries put forth by the LSST astrophysicists and provided important insights about how architectural characteristics of these systems affect the performance of these techniques. In particular, our evaluation shows that the choice of a database configuration can significantly impact the performance of the object association process. We evaluated the queries using different FOV region densities, but we were mostly interested in the high density fields because of its closeness in density to the average LSST case. This case involves searching for approximately 30 K detections in approximately 30 million objects using spatial attributes. These objects need to be matched within 5-10 seconds. For brevity we will show the results from the hybrid (third) configuration described above, the reader is advised to go to [KKA+09]for the details of the work.

In our evaluation, the algorithms behaved differently on different database system configuration. On the Netezza system, one of our benchmarks performed poorly because of the overhead of creating disk-based tables on-demand for each of the spatial regions and the lack of support for stored procedures in the system, which resulted in overheads due to external scripts being treated as independent queries. On the MySQL system however, neither algorithm performs as well as it does on the first configuration because it does not execute a query in parallel and executes JOINs on the frontend. This drawback can be remedied to an extent by partitioning a query into a set of smaller queries and executing these sub-queries as a batch query. This would take advantage of the high-throughput oriented design of the second configuration. The third configuration enables different partitioning and replication of the catalog tables across a collection of the independent database system instances. Although query performance was enhanced for this configuration, the data preparation phase and distribution remains a challenge and adds to the time of the query.

Table 1 below shows the results from running the query on a 16 node cluster with data replicated on all nodes. The table also lists the three different important phases that data has to go through before getting the results. We measured the time for data movement and preparation since in some cases it could delay the query results and we believe this should not be ignored. These results show that we can satsfy the high speed matching requirements for the application.

Table 1: Cross match time using 16x1 strategy and OptZone

	Prepare	Transfer time	Query time	Total time
High	1.9 s	5.6 s	1.3 s	7 s

As our study demonstrates, realistic evaluations are key to understanding the performance of complex, data intensive applications. To that end, there is a need to develop benchmarks and realistic use cases in order to help evaluate data intensive algorithms and architectures. These benchmarks should include a detailed description of how the data will queried and used and should cover all nine of the research challenges previously mentioned.

Section 3) Relational Databases, On-Line Analytical Processing, and Non-Traditional Database Environments

Relational database management systems (RDBMS) are the standard, nearly ubiquitous, technology used to store and search categorical data. The relational model was first outlined by Cobb [Cobb70] and is based on relational algebra. Many books have been written on the design and use of relational databases and the interested reader is directed to [SKS10] for a more detailed overview. For our purposes, it is sufficient to know that in the relational model all concepts are mapped to a set of two dimensional tables, where the columns represent attributes of concept and the rows represent a specific instance of that concept, and queried through a standard language called SQL. Relational database implementations support definition of keys and indices to enforce integrity constraints and enable efficient querying of the data.

Beyond the ease with which these complex tasks can be performed, the popularity of databases in commercial environments can be traced to three guarantees they provide. The first is ensuring atomicity: a set of commands can be grouped into a single transaction that will either be completed in its entirety or not at all, under no circumstances will only part of the transaction be performed (i.e. the collection is treated as an atomic instruction). Second, multiple transactions are serializable: the result of executing a

set of transactions must correspond to a sequential execution of the transactions, even if the individual commands are performed concurrently. Finally, databases are highly fault tolerant, with checkpoints supporting long-running series of transactions by enabling rollbacks and restarts if the database is interrupted. These benefits of relational databases come at a cost though, throughput on these systems is typically far from the raw disk bandwidth due to locking and synchronous disk writes.

Traditional relational databases are extremely useful for applications that take advantage of their guarantees. However, they do have significant limitations as well. A database is typically designed to serve a specific function (e.g. tracking sales) and is isolated from other databases. As a result, it is not uncommon for an organization's data to be distributed across multiple databases – for example, for each store to keep its own sales data. Furthermore, the resulting record-centric view is often optimized for data ingest, making it less useful for higher-level data analysis and reporting. To enable multi-terabyte or petabyte database analysis alternative data management approaches, specifically data warehouses and data cubes, have been developed.

Data warehouses [AGS97, CU07] may still be stored within an RDBMS, but have several important differences from traditional transaction-oriented databases.

- They are updated on a regular schedule, not as transactions occur, and thus may not have the most current information. Whether they are updated hourly, daily, weekly, or monthly depends both on the number of transactions being processed and the reporting requirements.
- They are focused on summary information. Aggregations, for example sales by item by store by day, are often created to reduce the size of the database and enable faster query responses. While this removes access to the detailed records, these aggregations are usually all that is needed for reports and analysis.
- A star or snowflake schema [GLK99] is used to organized data around facts and specified dimensions (columns in the original database). By reducing the number of dimensions and organizing the data around them, it is easier to generate the summarizations and views of the data required to perform analysis or generate reports.
- Data warehouses often integrate information from multiple transactional databases, for example from each store within a chain, to provide a more comprehensive view.

Data warehouses have been popular in business for over a decade, and have resulted in some of the first multi-terabyte data sets. While integration of the underlying data should be a simple process, in practice, a significant data cleaning step is required to resolve semantic inconsistencies between the databases, (e.g. converting data when an update has not been consistently applied). Despite this additional cost, analysis over data warehouses has allowed businesses to better understand their data and become significantly more efficient.

Unfortunately, even with a star schema, it can be challenging to interact with data warehouses due to the volume and relational organization of the data. On-Line Analytical Processing (OLAP) [CCS93], otherwise known as data cubes, are a non-relational data structure in which the N-dimensional cube, derived from the dimensions of the associated star schema, is stored in a format that supports fast projections of data summaries onto a small number of dimensions. Conceptually, the cube stores metrics, such as counts, totals, or averages in an efficient, but sparse, N-dimensional matrix, a representation that is often several orders of magnitude smaller than the original database. Queries then project this data onto

a view comprised of a relatively small number of dimensions (typically 2-4), aggregating the information further. Identifying informative views is typically performed by an expert, however some research in user guided view discovery has been performed [JBC+09]. Because the projection can be efficiently computed, data cubes, through the associated interfaces, provide an interactive way to explore large data sets. Several vendors provide excellent graphical interfaces to data cubes, enabling drag-and-drop data selection tied to charts and maps of the data.

While relational databases are extremely important for managing data that requires atomicity and serializability, as much business information does, these benefits apply primarily to categorical data. The associated overhead and table-oriented structure makes RBDMS less than ideal for other types of information:

- **Temporal data.** Time is represented as a singular value (i.e. a timestamp) and as a result relativistic queries (e.g. age, 3 days ago) can be asked but are not directly supported by SQL. More complex temporal queries (e.g. identifying overlapping periods or sequences of events) can be challenging to perform in a relational environment.
- **Semi-structured data (XML).** Semi-structured data does not map well to a traditional relational format, although it can be represented using *Name-Value* pairs to map attributes to the associated entities. Unfortunately, this type of structure makes it hard to do complex searches across objects and usually requires a significant number of (costly) joins. If the information being represented is primarily structured, adding semi-structured tables to a standard relational format may be acceptable. Due to the popularity, and importance, of this type of data, many RDBMS vendors are providing the ability to read, store, and write semi-structured data in an optimized data structure.
- **Image / video data.** Images do not map well to the table structure of relational environments, and thus are usually treated as single, binary large objects (BLOBs). This type of information can be read, but not searched, compared, or otherwise analyzed within the database.
- **Text.** Unstructured data such as text data can be stored within a relational database, in a form similar to a BLOB. Simple keyword searching is typically supported and many vendors provide additional search capabilities. However, more complex analysis (e.g. entity extraction, version control, clustering) are not generally supported. Document management systems are evolving to address this need and will likely become integral components of future database systems.

Scientific data has also remained a significant challenge for relational database technology. In part, this is because the associated applications typically require extremely fast I/O and are not concerned about serializability or atomicity [MC99]. Additionally, a simulation typically maps its data to a small number of extremely large tables. Analysis would require these tables to be repeatedly searched, joined, and sorted, expensive operations even with indices. Nonetheless, there have been some clear examples where relational technology has been extremely beneficial to scientific domains. One of the best known examples, is Jim Gray's release of the Sloan Digital Sky Survey (SDSS) server using a relational database backend [SGT+02]. The SDSS is a digital map of the northern sky that has enabled detection of hundreds of millions of new objects. The initial SDSS project, concluded in 2005, generated several terabytes of data which was freely accessible to both astronomers and the public through a web-based query interface. Follow-on projects have brought the total data set size up to 49.5TB, with approximately half of that representing the raw data. The key to the success of this project was to define the specific queries that

need to be enabled and to develop an appropriate set of metadata. The queries focused the database design leading to a more efficient implementation than would be possible if the database was required to fully support *ad hoc* queries. The metadata provided additional semantic information (e.g. from post-processing the data, tracking provenance, and soliciting user annotations) about the data which were then stored and used to efficiently answer these queries.

Despite these limited successes, there remains a disconnect between what the relational data model offers and what scientific data analysis needs that hampers widespread adoption. Most modern experimental science makes extensive use of *sensor arrays* to measure whatever physical phenomenon is the subject of study, be it an astronomical star, a wild fire, or diseased tissue[1]. Each of these observable features are typically represented as co-located pixels or *objects* derived from pixel data. Multiple observations of the same phenomenon are often recorded, yielding *time series* -- in some cases with frequency exceeding hundreds of observations per second. In other words, data tends to have well defined neighborhoods, and is inherently *ordered* in both space and time. For these reasons, an *array data model*, with implicit ordering, and notions of 'adjacency' or 'neighborhood', is far more desirable than tabular tables offered by relational technology.

Further, due to many factors, including imperfect observing conditions, detector mis-alignments, hardware limitations and failures, data is typically sparse and imperfect (i.e. "noisy" and uncertain). To process such data, specialized, mathematically and algorithmically sophisticated processing methods are required. Expressing these sophisticated algorithms through rich-but-constraining SQL is often next to impossible –allowing scientists to "package" their algorithms into reusable *user defined functions* (UDFs), and executing these functions where the data resides is far more desirable.

To illustrate the reasoning behind the above claims, consider a few typical scientific analyses: pair-wise comparisons of observation or entity records, such as searching for near-neighbor entities; computing regional densities; or finding spatial patterns. A naïve implementation for these analysis requires $O(N^2)$ comparisons, and with N measured in billions or trillions this becomes too expensive and slow to be practical. A common solution to dealing with a large data set is to partition it and analyze each partition in parallel. This approach is particularly popular for non-correlated data sets, as evidenced by the popularity of the map/reduce paradigm. Scientific observational data, however, is *highly* correlated and thus a naïve partitioning into chunks and processing these chunks in parallel is not a viable option. For example, pairwise analyses near partition edges requires "adjacent" partitions, leading to unbearably large inter-node data transfers needed to compute results across the edges. An effective data "de-correlation" can be done at the expense of small data duplication, commonly called overlapping partitioning. Unfortunately, overlapping partitioning is not implemented in any existing off-the-shelf data management system.

To illustrate need for new data abstractions and associated interfaces, consider analyses that involve finding pairs of objects within "similar" time series: since measurements of different objects are typically taken at different times, some measurements are missing, and those not missing are uncertain. Expressing an algorithm that compares these imperfect time series through SQL is a daunting task. Consequently,

1 Similar challenges can be observed in certain types of commercial applications such as risk management systems in financial applications, analysis of web log data, deep sequencing analytics for drug discovery, or digital medical imaging analysis.

scientists tend to run data analyses using an extract-transform-load (ETL) approach: data is extracted from database, processed using a procedural language in custom application software, and results are pushed back to the database. This means data is *moved* to the computation – something that was works well on megabytes or gigabytes of data, but fails miserably when terabytes or petabytes of data are required. For large data sets, moving computation to data is highly preferred; and this is typically achieved through UDFs. While UDFs have been implemented in DBMSs for many years, they were never extensively used for scientific analyses.

As a result data-intensive applications typically perform extreme-scale data analytics by building systems from the "bare metal" up. In some cases, workflow management tools [LAB+06, LG05, ABB+03] have been used to simplify the process of organizing large-scale analytical systems. These tools work best when data collections involve large numbers of moderately-sized files and are not optimized for observational data sets that contain immense number of relatively small records. Analyses involving processing entire data sets are typically planned and scheduled as production jobs, prohibiting curiosity-driven, what-if, ad-hoc analysis and delaying scientific discoveries. For this reason, many data-intensive industrial users, in particular Internet companies, have turned to the map/reduce (MR) model. The MR approach works particularly well on uncorrelated and unstructured data sets: segments of data are randomly distributed (hashed) across the available nodes and processed in parallel. This approach is less well-suited for the type of spatial or temporal correlations described above. Additionally, the MR model leaves the record structure for the user to define in code[2].

This recent trend, observed in industry and underlined by MR, departs from traditional, highly centralized database architectures. Instead, a system is deployed over a network of computers, each with its own locally attached storage. Each compute/storage node runs a semi-autonomous instance of a database engine, providing communications, query processing and a local storage manager. All instances share access to a centralized system catalog database (which could be logical) that stores information about the nodes, data distribution, user-defined extensions, and so on. This architecture allows for incremental (horizontal) upgrades of the system and provides dramatically better resilience against failures, as well as simplifying recovery.

SciDB [CKL+09, SBD+09], is one example of a new breed of database management systems designed to overcome these shortcomings and extend the benefits of database management systems to a broader community [3]. SciDB is an open source system currently under construction focused on addressing the challenges present in scientific analytics on highly dimensional, correlated, large-scale data sets. SciDB is not a traditional database: it is not optimized for online transaction processing (OLTP) and only minimally supports transactions at all; it does not have a rigidly-defined, difficult-to-modify schema. Instead, everything is designed to support analytics. Storage is write-once, read-many with bulk loading, rather than single-row inserts, as the primary input method. Functions and procedures can execute in parallel, as close to the data being operated on as possible. SciDB natively supports an array data model and query language with facilities that allow users to extend the system through new scalar data types and array operators.

---

2 This has been addressed to some extend by implementing limited RDBMS functionality on top of MR, including Hive and HadoopDB

[3] See [ABH09] for additional references on column-oriented databases

Section 4) Metadata and Provenance Management

Metadata captures and provides information about other, often digital objects. This information can include details such as:

- Topic - *Keywords* describing the domain the object belongs to.
- Description – *natural language comment about the object.*
- Access Conditions - information concerning who and how the object can be accessed
- Structure - *Description* of the organization (physical or virtual) of the object
- Content – *Key Parameters* distinguishing and characterizing the content of the object
- Location - *Navigational information* to where the object can be found (often geospatial in nature)
- Related Material - *References* to other objects, into the literature and community providing context about the object**.**

In data intensive computing, metadata is of particular importance because it can be used tohelp to organize and characterize the data andto enable effective discovery, access and analysis of relevant data. In addition it can be used to help to document important links between different data sets, scholarly publication or other sources of complimentary information, providing vital background and guidance for the exploration in particular in data intensive environment.

Repeatability, reproducibility and transparency are qualities that are at the heart of good research practices. In data intensive environments with high levels of distribution of data and/or compute resources, complex computational analysis and evaluation pipelines and processes, these qualities become more important. In order to support repeatability, reproducability and transparency in such complex settings, it is important to provide for the automated capture of information about the data creation process as well as information concerning potential later additional analysis and manipulations. Provenance metadata provides the history of ownership and creation of an object, and thus is a well-suited means to capture, manage and make accessible this kind of information to enable future use and reuse with confidence. The analysis of the provenance metadata can help to ascertain data quality and the rigor of the research process, establishing the data's 'pedigree' in both human and machine readable form. In addition, an increasing number of data generators are interested in having their authorship and affiliation associated with their data. Provenance metadata can help to provide attribution information along with data thus helping data generators begin to gain and maintain credit for their contributions.

Metadata provenance information is best captured in a structured well understood format so that programs and humans that access the information can understand what it means. It is also important for the metadata to be accessible. Thus a preferred solution is often to store it in what appears to be a centralized repository that is web accessible. Such a repository may truly be centralized or it may just have interface options that support centralalized search and access over what may be a distributed back end.
While metadata and provenance are critical across a broad range of settings, in this chapter we focus on how effective metadata and provenance management plays an essential role in data intensive simulations and analysis. Given this discussion, the relevance and applicability to other areas should become apparent.

Support for data-intensive application execution

Researchers today rarely engage any longer directly with their research object, but do so via digitally captured, reduced, calibrated, analyzed, synthesized and visualized data. Often at least some of the data is generated by other researchers and then reused. Advances in experimental and computational technologies have led to an exponential growth in the volumes, variety and complexity of this data, resulting in a growing number of petabyte and soon exabyte scale collections, predicted to outstrip the volumes of data available on the internet [D08]. Leading examples of data intensive simulations are:

- Climate models, expected to reach exascale by 2020 (*Challenges in Climate Change Science and the Role of Computing at the Extreme Scale*, http://extremecomputing.labworks.org/climate/report.stm)
- Computational astrophysics - e.g. predictions for the gravitational lensing signal – could reach 20 petabytes per simulation in 2012 (today 0.03 petabytes)
- New experimental facilities, for example the Large Synoptic Survey Telescope (LSST) will produce a multi-petabyte data set that will include 10 billion galaxies and a similar number of stars, energy research such as the European XFEL, will require near real-time analysis of experimental results produced at multi-petabytes per day from 2014 (today three petabytes per year at comparable facilities)
- The US Department of Defense is expecting yottabytes (10^{24}) of sensor data by 2015
- Internet services such as Google, Yahoo and Microsoft have already reached data volumes of 100's of petabytes.

A wide range of supporting technologies is required to allow applications to effectively utilize these large data sets. Metadata plays an essential role in most of these enabling technologies, making them effective in reducing the data space and volume, as well as enabling them to exploit relevant relationships between different, often heterogeneous and potentially distributed data objects. Challenging areas of data intensive computing support, where metadata and provenance have a crucial impact include: data identification and assessment, data sub-setting, monitoring and control of the simulation, provenance for simulation, data management for distributed applications and the annotation of the results for future analysis, reuse, and attribution. We explore each of these in turn below.

Data Identification and Assessment –With exponentionally growing data diversity and volumes, the selection of the most suitable data for any task at hand can be time consuming. Appropriate metadata and metadata systems can cut efforts dramatically, providing easily searchable summary information on provenance, content, quality and access conditions. Depending on the extent of the metadata, it can be possible to make very detailed assessments and comparisons between data sets, without the need to download and access the data directly. This is because provenance and key parameters from the data have been captured in the metadata systems and can themselves be analyzed. An example is:
. "*give me only data sets where value Z is in the range of F-G produced by radar A, with positions a Lat c – long D, at altitude X, which worked 100% during the selected period, and shows high quality verified data, -* ". In data rich environments metadata is fundamental to this kind of automated processing, identifying relevant data effectively and efficiently. One emerging trend in virtual observatories is to use

exactly this strategy to support two levels of queries – one identifying sources that contain data of interest and another query to actually obtain (and often plot) data (e.g.,VSTO [MFC07]). In contrast however to the World Wide Web, only selected scientific communities offer this type of identification and assessment support across organizational boundaries [SAB09], [LLM09], [MSF09].

Data Sub-setting –. It is very common that only a subset of a particular data file is required to initialize a simulation, and the simulation might need similar subsets from a collection of files or databases. In less data rich environments, with smaller data volumes, it is at times still possible to download the complete data object and filter out the required data during preprocessing, however in data intensive applications the volumes become prohibitive. For these applications, remote pre-selection and filtering is required to reduce the data to only the neccessary subset, thus reducing the volume that needs to be transferred. Metadata guided identification, assessment and selection reduces the volume of data to be accessed. Further metadata describing the structure, specific content and coverage of each data file allows algorithms to identify and extract the specific sub-set of the file(s) or databases that the user requires. Some communities have taken the concept further and provide the capabilities to:

- Capture specific data requirements of an applications through metadata
- Drive re-packaging of the data by matching data file and application metadata information (incl. execution environment)
- Deliver the right data to the right place for the simulation run e.g. in the cloud, on a campus grid computing environment, or on a leadership class computing facility requiring parallel I/O.

Metadata makes it possible here to create and guide automated processes that can respond flexibly to changing user requirements.

Monitoring and Control of the Simulation – Complex simulations require coupling simulation models that span different domains and different levels of theory, time, and scale to provide increased simulation fidelity. Computing environments themselves are also increasing in complexity with distributed data sources, leadership class computer systems, grid and cloud computing. The factors combined make it increasingly difficult to manage, monitor and control the execution of a particular simulation run or even more so combinatorial or ensemble rund with 10's – 100's of runs. Computational workflows and accompanying dashboard applications [A08] can support the automation and monitoring of single applications, and do so usually based on the simulation output. However, for data-intensive applications the volume of the output makes it very difficult to quickly identify the relevant information directly from the outputs. The incorporation of metadata can make it more efficient to capture and track the progress of applications. Meta data can be used to capture key derived values from simulation runs, enabling a quick assessment of application progress and correctness without the need to analyse voluminous simulation output files. . By collecting and analyzing this metadata not only for single runs, but also across a larger number of simulations, metadata can help identify both important trends in the scientific results and areas of operational concern. If key parameters of the simulation run have been collected, simple analysis or visualization programs can be used to give an overview of the result spread or trends across potentially thousands of runs, allowing the user to hone in on the important executions, without the need to analyze all resulting data sets in full [WBD09]. Similarly potential problem areas and trends in the code can be

easily identified and tracked, some of which would otherwise be impossible to pin down. Scientists who have extensively collected this type of metadata have reported that they have been able to improve the quality and accuracy of their code significantly [KJW10] due to the ability to track both intermittent errors and to spot trends in their calculations. Scientists found that with the support of provenance metadata they were able to carry out and analyze many more simulations in a shorter time, leading to greater accuracy of their results, due to more complete information. Again metadata is used to identify relevant information more quickly, aiding the monitoring, analysis and improvement of applications and simulation processes. If stored, this information also forms a critical part of the provenance of a simulation run, allowing users to assess a workflow execution after its execution.

Provenance for Applications – With the increase in data and complexity of compute environments, there has been a corresponding increase in the number of applications carried out (e.g., . for scenario analysis, parameter sweeps, ensemble runs, distributed applications or large scale collaborative projects). In each of these scenarios it is of fundamental importance for the assessment of the results to have information available on how the results were obtained i.e.:

- Which code and code versions were used?
- What was the input set and code configuration?
- Where and how was it compiled and executed?
- Were there any special occurrences such as restarts, warnings or errors, etc. ?

Provenance metadata is designed to capture and make this type of information available; ideally any such provenance information should be captured automatically as it becomes available to ensure ease of use for the scientists and reliable quality and accuracy of the data. This metadata enables scientits to retrace an execution, and for example verify if some unexpected findings are potentially caused by a problem in the code or input values. Further, at the time of execution additional information such as key parameters in the output could be captured.

Data Management for distributed applications – A particular class of data-intensive applications run in highly distributed environments, where data and/or computing are found in many different locations. Examples are climate simulations driven by a variety of observational data reaching from NASA satelite data to sensor network measurements, or meta genomics analysis in biology, which analysis results from many different experimental results. Given the volumes of data involved in data intensive applications, it is not feasible to move all data into a single location, either before analysis or afterwards. Therefore it is paramount to support applications in locating and management of their data, in particular for highly distributed applications. Metadata available for input data locations and captured during the application run can help to track the location of input and output data for particular simulations or related to particular simulation components. It can provide an easy to use logical link to each data object, independent of its physical location to enable access to the data at any stage. In addition the metadata can capture detailed information about the actual physical location of each data object and the organization of any compound data set. It can be used to support services to resolve the logical link to the data used by applications into the data's actual physical location and accompanying access mechanism at any given point in time. This information can be used both during and after the simulation run to coordinate access to the data, support

scheduling of computational tasks close to the data e.g. for further analysis, data exchange and final data collection for archival if required. This type of metadata can also be used to add an additional layer of monitoring, control and audit for complex data movement processes.

Annotation of results for analysis and future reuse – Closing the circle, with the large volumes of data produced by data-intensive application codes it is paramount to annotate the results appropriately to allow for further analysis and potential future reuse of the data by students, collaborators and the wider scientific community. In this step a range of different metadata components are combined, including those described in the previous sections as well as topic, access conditions, content and related materials, to form a full descriptive record.

To make the capture of the metadata and provenance information practical and reliable it is paramount to automate most of the processes involved in its capture, management and analysis. Furthermore these metadata -related processes are best tightly integrated into computing environments, as studies have shown that the usage of metadata is most beneficial and transformational when seamlessly integrated into the research process. An excellent example is the US Earth Systems Grid [SAB09].

US Earth Systems Grid (ESG) – This project provides scientists with access to essential climate data from worldwide distributed resources, as well as analysis capabilities. It uses community agreed data and metadata formats and standards to describe data content and structure. International resources are linked in through mappings between their own data formats and the ESG formats, giving scientists seamless access to all resources. To allow users to explore the large number and size of the different data holding the project has developed a multi-tier metadata based approach for data discovery and assessment. Further, standardized services allow the user to select particular sub-sets of the data to minimize the size of data that needs to be transported; these subset definitions are based on standardized metadata description of the data structure and content. With growing data sizes, this is often not enough and the ESG also offers remote analysis capabilities, whereby a copy of the required data nearest to a suitable computing resource is identified and efficiently moved with special protocols; metadata records help to identify suitable data replicates worldwide. The results of climate simulations are automatically annotated and archived where desired. The infrastructure has been successfully used for a number of international climate inter-comparison studies, due to the accessibility and value of the results. Due to the extensive metadata, the climate community has been able to analyze the data effectively, resulting in many hundreds of peer reviewed scientific papers and contributing to the Nobel Prize-winning IPCC Fourth Assessment Report (AR4).

Support for data analysis and exploitation in data intensive environments

The analysis of extreme scale data has many challenges, key are those of moving large amounts of data and the identification of relevant information in large volumes of data. Metadata can be of essential help in addressing both of these challenges through support of: efficient search and retrieval, capture of key parameters, feature identification and extraction, inference support. As the significant reduction of search effort through descriptive provenance metadata and availability of key parameters for data files, has already been described we will focus here on feature identification and inference support, which are of particular benefit for the data-intensive analysis of extreme scale data.

Feature identification and cataloguing – When assessing and analyzing the output of application runs or experimental/observational studies, researchers often are not so much interested in the data as a whole (statistical analysis), or the developments of particular variables over time or space, but are aiming to identify particular complex phenomena – features- within the data, such as: a dying star, a tornado or a group of suspect individuals communicating. Currently much of this identification work has to be done by visually tracing through large amounts of data. In data intensive computing environments, this task can no longer be accomplished by a single researcher or group. Instead, some are exploiting social data analysis [FS08], [GW09], where the general public is encouraged to look through visualizations or run small analysis on their systems to identify particular phenomena in the data and contribute results back to the repository. To make the most of the effort expended in identifying the various features in the data, researchers want to catalogue these described by metadata, so that they and others such as collaborators can find and access them again quickly for further analysis. Formalizing such metadata descriptions of phenomena and features will allow the identification, extraction and comparison of such events across different sources examples for such systems found in particular in biology, astronomy, climate and earth systems communities [HBC05], [WLL06].

Next to quantitative and qualitative analysis of such events enabled through advanced access to features in the data, this also supports ongoing work further characterizing such events and eventually enabling the automated detection of such features in the data, a development which, if it can be used with confidence in its accuracy, would speed up research processes in data rich environments tremendously. Again metadata is instrumental here in providing efficient access to the relevant data, increasing the accuracy of methods through access to more complete data.

Inference Support – Given the volume of data generated in data rich environments it is important to develop efficient analysis methods to enable researchers to explore the information and knowledge contained in the raw data. Advanced analysis methods and their support through metadata and provenance data are essential to identify features and patterns of interest in the data, semantic inference support builds on these and aims to draw new conclusions from data that have not been explicitly expressed in the data itself. Deductive inference can be supported using existing reasoners providing standard inference rule chaining. Biology is a key scientific discipline which utilizes this technique, e.g. the discovery of regulatory gene networks, which encode gene function, can be aided by using prior biological knowledge captured through ontologies to infer likely function in combination with the analysis of experimental results [SBB09]In the latter the meaning contained in the metadata is encoded in mathematical terms (vectors in highly dimensional space), a closeness of concepts can be deduced through distance calculations (between vectors), probabilities through analysis of clustering of concepts (vectors).

Particular metadata challenges in data-intensive computing environments

Just as access and management of data presents specific challenges in data intensive computing environments, metadata faces its own set of hurdles. These include the rate at which metadata needs to be created and captured, the appropriate granularity for metadata annotation, the volume of metadata potentially created, and the overhead that the creation of metadata might introduce into an application. As data is produced at ever faster rates and volumes, the rate and volume of the metadata increases too, requiring the optimization and intelligent design of the underlying metadata systems to keep a pace with the increased demand. The emphasis will be placed on efficient metadata systems which may mean

compact metadata formats (e.g. non XML), efficient syntax (capturing as much information as required in the least verbose way), optimized access options, and/or partitioning strategies. In some cases, where the volume of metadata grows significantly, it might also be necessary to review if the level of detail captured is still appropriate or if reduced summary information, along with optional followup systems that may gather more provenance information only when requested, would be better. Finally, optimized database structures and connection mechanisms (such as connection pooling) may need to be employed to provide both the necessary speed and volume.

As the world is becoming more data rich and connected, the ability to focus quickly not only on the relevant data, but particular features within them becomes crucial. As more data is being processed by ever more complex processes it is essential that is possible to verify these workflows, so that the resulting outcomes can be used with confidence. Without metadata and provenance none of this could be accomplished, making it a fundamental enabling technology for data intensive computing.

Bibliography

[A08] Ilkay Altintas, "Lifecycle of Scientific Workflows and their Provenance: A Usage Perspective," Services, IEEE Congress on, pp. 474-475, 2008 IEEE Congress on Services - Part I, 2008

[ABB+03] Altintas I., S. Bhagwanani, D. Buttler, S. Chandra, Z. Cheng, M. Coleman, T. Critchlow, A. Gupta, W. Han, L. Liu, B. Ludaescher, C. Pu, R. Moore, A. Shoshani, M. Vouk, "A Modeling and Execution Environment for Distributed Scientific Workflows," Proc. 15th IEEE International Conference on Scientific and Statisical Database Management (SSDBM 2003).

[ABH09] D.J. Adabi, P.A. Boncz, S. Harizopoulos, "Column-oriented Database Systems", Proceedings of the VLDB Endowment, archive Volume 2 , Issue 2 (August 2009) Pages: 1664-1665, 2009 ISSN:2150-8097. http://cs-www.cs.yale.edu/homes/dna/papers/columnstore-tutorial.pdf

[ACI09] Nawab Ali, Philip Carns, Kamil Iskra, Dries Kimpe, Samuel Lang, Robert Latham, Robert Ross, Lee Ward, and P. Sadayappan. Scalable I/O Forwarding Framework for High-Performance Computing Systems, IEEE International Conference on Cluster Computing (Cluster 2009), New Orleans, LA, September 2009.

[AGS97] Agrawal, R., Gupta, A., Sarawagi, S. "Modeling Multidimensional Databases". In Proc. 13th Int. Conf. on Data Engineering (1997)

[BLMNA07] J. Becla, K.-T. Lim, S. Monkewitz, M. Nieto-Santisteban, A. Thakar, Organizing the extremely large LSST database for real-time astronomical processing17th Annual Astronomical Data Analysis Software and Systems Conference (ADASS 2007), London, England.

[BMF07] James L. Benedict, Deborah L. McGuinness, and Peter Fox. A Semantic Web-based Methodology for Building Conceptual Models of Scientific Information. In American Geophysical Union, Fall Meeting (AGU2006), San Francisco, Ca., December, 2007.

[CDG+06] F. Chang, J. Dean, S. Ghemawat, W.C. Hsieh, D.A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R.E. Gruber. Bigtable: A distributed storage system for structured data 7th USENIX Symposium on Operating Systems Design and Implementation, 2006.

[CCS93] Codd, E.F., Codd, S.B., Salley, C.T. "Providing OLAP (On-Line Analitycal Processing) to User-Analysts" An IT Mandate" (1993), www.cs.bgu.ac.il/~dbm031/dw042/Papers/olap_to_useranalysts_wp.pdf

[CHA10] Philip Carns, Kevin Harms, William Allcock, Samuel Lang, Robert Latham, and Robert Ross. Storage Access Characteristics of Computational Science Applications, in Proceedings of Supercomputing, New Orleans, LA, November, 2010. (under review).

[CKL09] P. Cudre-Mauroux, H. Kimura, K.-T. Lim, J. Rogers, R. Simakov, E. Soroush, P. Velikhov, D.L. Wang, M. Balazinska, J. Becla, D. DeWitt, B. Heath, D. Maier, S. Madden, J. Patel, M. Stonebraker, S. Zdonik, "A Demonstration of SciDB: A Science-Oriented DBMS", VLDB'09 Volume 2, Number 1, 1534-1537, Lyon, France, August 2009 http://www.vldb.org/pvldb/2/vldb09-76.pdf

[CLR09] P. Carns, S. Lang, R. Ross, M. Vilayannur, J. Kunkel, and T. Ludwig, Small-file access in parallel file systems. In Proceedings of the 23rd IEEE International Parallel and Distributed Processing Symposium, April 2009.

[Codd70] E.F. Codd, "A relational Model for Large Shared Data Banks". Communications of the ACM, Vol. 13, Num. 6, June 1970

[CSG99] D. Culler, J. Singh, A. Gupta. Parallel Computer Architecture: A Hardware/Software Approach. Morgan Kaufmann, San Francisco, CA, 1999.

[CU97] Chaudhuri, S., Umeshwar, D.: An Overview of DataWarehousing and OLAP Technology. ACM SIGMOD Record 26(1), 65–74 (1997)

[D08] Department of Defense, JASON Defense Advisory Panel Report, Data Analysis Challenges, JSR-08-142, December 2008

[DER06] Frank Dehne, Todd Eavis and Andrew Rau-Chaplin, The cgmCUBE project: Optimizing parallel data cube generation for ROLAP, Journal of Distributed and Parallel Databases 19 (1) (2006), pp. 29–62.

[FS08] Freire J, Silva C. 2008. "Towards Enabling Social Analysis of Scientific Data" In Proceedings of *CHI Social Data Analysis Workshop,* 2008.

[GHL98] W. Gropp, S. Huss-Lederman, A. Lumsdaine, E. Lusk, B. Nitzberg, W. Saphir, M. Snir. MPI—The Complete Reference, Volume 2, The MPI Extensions. The MIT Press, Cambridge, MA,1998.

[GLK99] V Gopalkrishnan, Q Li, K Karlapalem, "Star/Snow-Flake Schema Driven Object-Relational Data Warehouse Design and Query Processing Strategies", Lecture Notes in Computer Science, Volume 1676/1999, Springer Berlin / Heidelberg, 1999

[GP] http://www.softdevtools.com/modules/news/article.php?storyid=1101

[GW09] Goodman AA, Wong CG. 2009. "Bringing the Night Sky closer: Discoveries in the Data Deluge" In *The Fourth Paradigm: Data-Intensive Scientific Discovery*,2006, Microsoft Research.

[GWG05] G. Grider, L. Ward, G. Gibson, R. Ross, R. Haskin, and B. Welch. POSIX I/O Extensions Workshop, Carnegie Mellon University, 2005.

[GSS] Jim Gray, Maria A. Nieto-Santisteban, ALex S. Szalay, The zones algorithm for finding points-near-a-point or cross-matching spatial datasets, The ACM Computing Research Repository (CoRR) abs/cs/0701171.

[HBC05] Jodi E. Hirschman, Rama Balakrishnan, Karen R. Christie, Maria C. Costanzo, Selina S. Dwight, Stacia R. Engel, Dianna G. Fisk, Eurie L. Hong, Michael S. Livstone, Robert Nash, Julie Park, Rose Oughtred, Marek Skrzypek, Barry Starr, Chandra L. Theesfeld, Jennifer Williams, Rey Andrada, Gail Binkley, Qing Dong, Christopher Lane, Stuart Miyasato, Anand Sethuraman, Mark Schroeder, Mayank K. Thanawala, Shuai Weng, Kara Dolinski, David Botstein, and J. Michael Cherry. Genome Snapshot: a new resource at the *Saccharomyces* Genome Database (SGD) presenting an overview of the *Saccharomyces cerevisiae* genome *Nucl. Acids Res. 34(suppl 1): D442-D445 doi:10.1093/nar/gkj117*

[HDF5] The Heirarchical Data Format, version 5 (HDF5). http://www.hdfgroup.org

[HHT05] M. Hadjieleftheriou, E. Hoel, V. J. Tsotras, Sail: A spatial index library for efficient application integration, GeoInformatica 9 (4) (2005) 367–389.

[HTT09] Tony Hey, Stewart Tansley, and Krist in Tolle. The Fourth Paradigm, Data-Intensive Scientific Discovery, Microsoft Research, Redmond, Washington, October 2009

[JBC+09] C. Joslyn, J. Burke, T. Critchlow, N. Hengartner, E. Hogan, "View Discovery in OLAP Databases through Statistical Combinatorial Optimization", in Proceedings of the 21st International Conference on Scientific and Statistical Database Management, New Orleans, LA, June 2009.

[KBB08] P. Kogge, K. Bergman, S. Borkar, D. Campbell, W. Carlson, W. Dally, M. Denneau, P. Franzon, W. Harrod, K. Hill, J. Hiller, S. Karp, S. Keckler, D. Klein, R. Lucas, M. Richards, A. Scarpelli, S. Scott, A. Snavely, T. Sterling, R. S. Williams, and K. Yelick. Exascale computing study: Technology challenges in achieving exascale systems. Technical report, DARPA, 2008.

[KJW10] K Kleese van Dam, M James, A Walker. *Integrating Data Management and Collaborative Sharing with Computational Science Research Processes*. In Handbook of Research on Computational Science and Engineering: Theory and Practice, IGI Global – in print, expected in September 2011

[KKA+09] Vijay S. Kumar, Tahsin Kurc, Ghaleb Abdulla, Scott R. Kohn, Joel Saltz, Celeste Matarazzo. Architectural Implications for Spatial Object Association Algorithms, In proc. of the IEEE International Parallel and Distributed Processing Symposium (IPDPS 2009).

[LAB+06] B. Ludaescher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, and Y. Zhao, "Scientific Workflow Management and the Kepler System," Concurrency and Computation: Practice & Experience, 2006.

[LCL09] S. Lang, P. Carns, R. Latham, R. Ross, K. Harms, and W. Allcock. I/O Performance Challenges at Leadership Scale. In SC '09: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis. New York, NY, November, 2009.

[LCR03] J. Li, W-K. Liao, A. Choudhary, R. Ross, R. Thakur, W. Gropp, and R. Latham. Parallel netCDF: A scientific high-performance I/O interface. Technical Report ANL/MCS-P1048-0503, Mathematics and Computer Science Division, Argonne National Laboratory, May 2003.

[LG05] B. Ludaescher and C. A. Goble, editors. ACM SIGMOD Record, Special Section on Scientific Workflows, volume 34(3), September 2005.

[LKS+08] J.F. Lofstead, S. Klasky, K. Schwan, N. Podhorszki, C. Jin, Flexible IO and Integration for Scientific Codes through the Adaptable IO System In Proc. International Workshop on Challenges of Large Applications in Distributed Environments, pp. 15-24, 2008.

[LLM09] B.N Lawrence, R Lowry, P Miller, H Snaith, and A Woolf. Information in environmental data grids. In Phil Trans R Soc A 2009 367: 1003-1014.

[LSST] http://www.lsst.org/lsst/science/concept_data

[MA05] Mohammed . F. Mokbel and Walid G. Aref. PLACE: A Scalable Location-aware Database Server for Spatio-temporal Data Streams. Data Engineering Bulletin, 28(3), 2005

[MC99] R. Musick, and T. Critchlow, "Practical Lessons in Supporting Large Scale Computational Science," SIGMOD Record, 28, (4), (December 1999).

[MFC07] Deborah McGuinness, Peter Fox, Luca Cinquini, Patrick West, Jose Garcia, James L. Benedict, and Don Middleton. The Virtual Solar-Terrestrial Observatory: A Deployed Semantic Web Application Case Study for Scientific Research. In the Proceedings of the Nineteenth Conference on Innovative Applications of Artificial Intelligence (IAAI-07).

Vancouver, British Columbia, Canada, July 22-26, 2007. http://www.ksl.stanford.edu/KSL_Abstracts/KSL-07-01.html

[MGA03] Mohamed F. Mokbel, Thanaa M. Ghanem, and Walid G. Aref. Spatio-temporal Access Methods. IEEE Data Engineering Bulletin, 26(2), 2003.

[MSF09] B Matthews, S Sufi, D Flannery, L Lerusse, T Griffin, M Gleaves and K Kleese van Dam. Using a Core Scientific Metadata Model in Large-Scale Facilities. In 5th International Digital Curation Conference (IDCC 2009), London, UK

[MRG+05] Alan M. Maceachren, Anthony Robinson, Steven Gardner, Robert Murray, Mark Gahegan, Elisabeth Hetzler, Visualizing geospatial information uncertainty: What we know and what we need to know. Cartography and Geographic, Journal of Information Science, (32) 2005, page 160

[NET] http://www.netezza.com/

[NKK+08] M. Nanni, B. Kuijpers, C. Korner, M. May, and D. Pedreschi. Spatiotemporal Data Mining. In F. Giannotti and D. Pedreschi, editors, Mobility, Data Mining, and Privacy: Geographic Knoweledge Discovery. Springer-Verlag, 2008.

[OR] Oracle TimesTen In-Memory Database Architectural Overview Release 6.0, http://www.oracle.com/database/timesten.html

[POS96] IEEE/ANSI Standard. 1003.1 Portable Operating System Interface (POSIX) - Part 1: System Application Program Interface (API) [C Language], 1996.

[PW98] Paul Werstein, A Performance Benchmark for Spatiotemporal Databases In Proc. Annual Colloquium of the Spatial Information Research Centre, pp. 365–373, 1998.

[S99] Timos Sellis, Research Issues in Spatio-Temporal Database Systems, R.H. Güting, D. Papadias, F. Lochovsky (Eds.): SSD'99, LNCS 1651, pp. 5-11, 1999, Springer-Verlag Berlin Heidelberg 1999

[S03] P. Schwan. Lustre: Building a file system for 1000-node clusters. In Proceedings of the 2003 Linux Symposium, Ottowa, Canada, July, 2004.

[S08] Markus Schneider. Fuzzy Spatial Data Types for Spatial Uncertainty Management in Databases. Handbook of Research on Fuzzy Information Processing in Databases. Information Science Reference, 490-515, 2008

[SAB09] F. Siebenlist, R. Ananthakrishnan, D.E. Bernholdt, L. Cinquini, I.T. Foster, D.E. Middleton, N. Miller, and D.N. Williams. Enhancing the earth system grid security infrastructure through single sign-on and autoprovisioning. In Proceedings of the 5th Grid Computing Environments Workshop (Portland, Oregon, November 20 - 20, 2009). GCE '09. ACM, New York, NY, 1-8. DOI= http://doi.acm.org/10.1145/1658260.1658278

[SBD+09] M. Stonebraker, J. Becla, D. DeWitt, K-T. Lim, D. Maier, O. Ratzesberger, S. Zdonik, Requirements for Science Data Bases and SciDB, CIDR 2009 Conference, Asilomar, CA, USA, January 2009 http://www-db.cs.wisc.edu/cidr/cidr2009/Paper_26.pdf

[SEN10] S. Shepler, M. Eisler, D. Noveck. Network File System (NFS) Version 4 Minor Version 1 Protocol. January, 2010. http://datatracker.ietf.org/doc/rfc5661/.

[SH02] F. Schmuck and R. Haskin, GPFS: A shared-disk file system for large computing clusters. in Proceedings of the FAST 2002 Conference on File and Storage Technologies, January 2002.

[SKS10] A. Silberschatz, H. Korth, S. Sudarshan, "Database Systems Concepts". MgGraw-Hill Publishing, January 2010

[SGT+02] Alexander S. Szalay, Jim Gray, Ani Thakar, Peter Z. Kunszt, Tanu Malik, Jordan Raddick, Christopher Stoughton, Jan vandenBerg: The SDSS skyserver: public access to the sloan digital sky server data. SIGMOD Conference 2002: 570-581

[SWLW03] Wenzhong SHI, Shuliang WANG, Deren LI, Xinzhou WANG, Uncertainty-Based Spatial Datamining, ASIAGIS, 2003

[TGL99] R. Thakur, W. Gropp, and E. Lusk. Data sieving and collective I/O in ROMIO. In the Proceedings of the Seventh Symposium on the Frontiers of Massively Parallel Computation, 1999.

[TGLU99] R. Thakur, W. Gropp, and E. Lusk. On implementing MPI-IO portably and with high performance. In Proceedings of the 6th Workshop on I/O in Parallel and Distributed Systems. ACM Press, May 1999.

[TOP509] Top500 List, November, 2009. http://www.top500.org/list/2009/11/100.

[TPRB03] M. Twa, S. Parthasarathy, T. Rosche, and M. Bullmer. Decision tree classification of spatial data patterns from videokeratography using zernike polynomials. SIAM International Conference on Data Mining, 2003.

[WBD09] A.M Walker, R.P Bruin, M.T Dove, T.O.H White, K Kleese van Dam and R.P Tyer. Integrating computing, data and collaboration grids: the RMCS tool Phil. Trans. R. Soc. A March 13, 2009 367 (1890) 1047-1050; doi:10.1098/rsta.2008.0159

[WLL06] A.Woolf, B. Lawrence, R. Lowry, K. Kleese van Dam, R. Cramer, M. Gutierrez, S. Kondapalli, S. Latham, D. Lowe, K. O'Neill, and A. Stephens. Data integration with the Climate Science Modeling Language. Adv. Geosci., 8, 83–90, 2006, www.adv-geosci.net/8/83/2006/

[WUA08] B. Welch, M. Unangst, Z. Abbasi, G. Gibson, B. Mueller, J. Small, J. Zelenka, and B. Zhou. Scalable performance of the Panasas parallel file system, in Proceedings of the 6th USENIX Conference on File and Storage Technologies, February 2008.

[XHL90] X. Xu, J. Han, W. Lu: "RTtree: An Improved Rtree Index Structure for Spatiotemporal Databases", Proceedings of the 4th International Symposium on Spatial Data Handling (SDH), 1990.