GRID DATA STREAMING

Wen Zhang[1], Junwei Cao[2,3*], Lianchen Liu[1,3], and Cheng Wu[1,3]

[1]Department of Automation, Tsinghua University, Beijing China

[2]Research Institute of Information Technology, Tsinghua University, Beijing China

[3]Tsinghua National Laboratory for Information Science and Technology, Beijing China

Abstract

Flourish development of grid computing have been seen in recent years which has enabled researchers to collaborate more efficiently by sharing computing, storage, and instrumental resources. Data grids focusing on large-scale data sharing and processing are most popular and data management is one of most essential functionalities in a grid software infrastructure. Applications such as astronomical observations, large-scale numerical simulation, and sensor networks generate more and more data, which constitutes great challenges to storage and processing capabilities. Most of these data intensive applications can be considered as data stream processing with fixed processing patterns and periodical looping. Grid data streaming management is gaining more and more attention in the grid community. In this work, a detailed survey of current grid data streaming research efforts is provided and features of corresponding implementations are summarized.

While traditional grid data management systems provide functions like data transfers, placements and locating, data streaming in a grid environment requires additional supports, e.g. data cleanup and transfer scheduling. For example, at storage-constraint grid nodes, data can be streamed, made available to corresponding applications in an on-demand manner, and finally cleaned up after processing is completed. Grid data streaming management is particularly essential to enable grid applications on CPU-rich but storage-limit grid nodes. In this work, a grid data streaming environment is proposed with detailed system analysis and design. Several additional modules, e.g. performance sensors, predictors and schedulers, are implemented. Initial experimental results show that data streaming leads to a better utilization of data storage and improves system performance significantly.

Key Words—Grid computing, data streams, and data streaming applications.

---

* E-mail: jcao@tsinghua.edu.cn. This work is funded by the Ministry of Education of China under the quality engineering program for higher education and the Ministry of Science and Technology of China under the national 863 high-tech R&D program (grant No. 2006AA10Z237).

1 Introduction

Enabling more efficient collaboration by sharing computing, storage, and instrumental resources, the Grid [1] has changed the manners in which scientists and engineers carry out their research by supporting more complex forms of analysis, facilitating global collaboration and addressing global challenges (e.g. global warming, genome research etc).

The terminology of the Grid is derived from power grids that have been proved to be successful. The Grid is a common infrastructure of services to enable cross-domain distributed computing. It hides the heterogeneity of geographically distributed resources, such as supercomputers, storage resources, networks, and expensive instruments, and makes them collaborate smoothly by scheduling jobs and workflows, moving data objects, monitoring and recovering from errors and failures. The Grid emphases on security, authentication and authorization issues to enable a Virtual Organization (VO) for problem solving.

According to their functions and characteristics, there are different types of grids, such as Computational Grids, which facilitate scalable high performance computing resources (e.g. large scale simulations); Data Grids [3][45], focusing on large-scale data sharing and processing held in multiple locations and also kept in multiple replicas for high availability [4][7][29][30][46]; Access Grids [61], enabling advanced video conferencing to facilitate collaboration between researchers nationally and internationally; and Sensor Grids, which collect real time data (e.g. traffic flows and electronic transactions).

Grid technology will play an essential role in constructing worldwide data analysis environments where thousands of scientists and engineers will collaborate and compete to deal with the challenges in our world, where high performance and data-intensive, or data-flow driven computing and networking technologies have become a vital part of large-scale scientific research projects in areas such as high energy physics, astronomy, space exploration and human genome projects. One example is the Large Hadron Collider (LHC) [19] project at CERN, where four major experiment groups will generate an order of Petabytes of raw data from four big underground particle detectors each year.

Data streaming applications, as a novel form of data analysis and processing, has gained wide interest in the community of scientists and engineers. Instruments and simulations are generating more and more data every day, which baffles even the storage with the largest capacities. Fortunately, what is concerned most is the information concealed in the raw data, so not all data requires storage for processing. Data streaming applications enable us to retrieve information we care about when data streams by.

Regular database management systems (DBMS) store tables with limited sizes. While stream database systems (SDBS) deal with on-line streams with unlimited sizes, data streams have specifics that requires different handling from DBMS. A lot of research in stream data management has been done recently [14][23][34][38][48] and the area offers a number of open research challenges. Several important characteristics of data streams make them different from other data: they are infinite, once a data element has arrived, it is processed and either archived or removed, i.e. only a short history can be stored in the database. It is also preferable to process data elements in the order they arrive, since sorting, even of sub-streams of a limited size, is a blocking operation.

Using Grid technology to address challenges of data stream applications gives birth to a new manner of data analysis and processing, namely, Grid data streaming, which will be elaborated in later sessions.

This article is organized as follows: Section 2 introduces basic concepts of Grid data streaming as an overview; relevant techniques are discussed in Section 3, and the next section describes popular applications of Grid data streaming. There are still some open research issues which are summarized in Section 5 together with our proposal on on-demand data streaming and just-in-time data cleanup. The last section concludes the whole article with a brief introduction to future work.

2 Overview of Grid data streaming

Grid computing has made a major impact in the field of collaboration and resource sharing, and data Grids are generating a tremendous amount of excitement in Grid computing. Now, Grid computing is promising as the dominant paradigm for wide-area high-performance distributed computing. As some Grid middleware, such as Globus [40], have been put into practice, they are enabling a new generation of scientific and engineering application formulations based on seamless interactions and coupling between geographically distributed computation, data and information services. A key quality of service (QoS) requirement of these applications is the support for high throughput and low latency data streaming between corresponding distributed components. A typical Grid-based fusion simulation workflow is demonstrated in [49], which consists of coupled simulation codes running simultaneously on separate High Performance Computing (HPC) resources at supercomputing centers. It must interact at runtime with services for interactive data monitoring, online data analysis and visualization, data archiving and collaboration that also run simultaneously on remote sites. The fusion codes generate large amount of data, which must be streamed efficiently and effectively between these distributed components. Moreover, data streaming services themselves must have minimal impact on the execution of simulation, satisfy stringent application/user space and time constraints, and guarantee that no data is lost.

Definition of a stream is data that is produced while a program is running, for example, standard out, standard errors or any other output files, which are contrast with output files that are only available once a program has completed. Traditionally, static data are stored as finite, persistent data set, while data stream is a continuous sequence of ordered data, whose characteristics can be described as append only, rapid, not stored, time-varying and infinite. Put it in another way, data streams are indefinite sequence of events, messages, tuples and so on, which are often time marked, namely, they are often marked with time stamps. What's more, streams are usually generated at many locations, i.e., streams are distributed.

Characters of data streams are as follows:

- Different from finite, static data stored in flat files and database systems
- Transient data that passes through a system continuously
- Only one look –single linear scan algorithms
- Records arrive at a rapid rate
- Dynamically changes over time, perhaps fast changing
- Huge volumes of continuous data, potentially infinite
- Requiring fast real time response

- Data can be structured or unstructured

Advances in sensing capabilities, computing and communication infrastructures are paving the way for new and demanding applications, i.e., streaming applications. Such applications, for example, sensor networks, mobile devices, video-based surveillance, emergency response, disaster recovery, habitat monitoring, telepresence, and web logs, involve real-time streams of information that need to be transported in a dynamic, high-performance, reliable and secure fashion. These applications in their full form are capable of stressing the available computing and communication infrastructures to their limits. Streaming applications have the following characteristics: 1) they are continuous in nature, 2) they require efficient transport of data from/to distributed sources/sinks, and 3) they require the efficient use of high-performance computing resources to carry out computing-intensive tasks in a timely manner.

To cope with the challenges put forward with analysis and processing of data streams, just like the database management system (DBMS), some data stream management systems, DSMS in short, are developed. Some prototypes are summarized as follows:

Researchers in Stanford University developed a general-purpose DSMS, called the STanford stREam dAta Manager (STREAM) [42], for processing continuous queries over multiple continuous data streams and stored relations. STREAM consists of several components: the incoming Input Streams, which produce data indefinitely and drive query processing; processing of continuous queries typically requires intermediate state, i.e., Scratch Store; an Archive, for preservation and possible offline processing of expensive analysis or mining queries; Continuous Queries [10], which remain active in the system until they are explicitly reregistered. Results of continuous queries are generally transmitted as output data streams, but they could also be relational results that are updated over time (similar to materialized views).

Aurora[36] is designed to better support monitoring applications, where streams of information, triggers, imprecise data, and real-time requirements are prevalent. Aurora data is assumed to come from a variety of data sources such as computer programs, sensors, instruments and so on. The basic job of Aurora is to process incoming streams in the way defined by an application administrator, using the popular boxes and arrows paradigm found in most process flow and workflow systems. Hence, tuples flow through a directed acyclic graph (DAG) of processing operations. Ultimately, output streams are presented to applications, which must be programmed to deal with the asynchronous tuples in an output stream. Aurora can also maintain historical storage, primarily in order to support ad-hoc queries.

Continuous queries (CQs) are persistent queries that allow users to receive new results when they become available, and they need to be able to support millions of queries. NiagaraCQ [23], the continuous query sub-system of the Niagara project, a net data management system being developed at University of Wisconsin and Oregon Graduate Institute, is aimed to addresses this problem by grouping CQs based on the observation that many web queries share similar structures. NiagaraCQ supports scalable continuous query processing over multiple, distributed XML files by deploying the incremental group optimization ideas. A number of other techniques are used to make NiagaraCQ scalable and efficient. 1) NiagaraCQ supports the incremental evaluation of continuous queries by considering only the changed portion of each updated XML file and not the entire file. 2) NiagaraCQ can monitor and detect data source changes using both push and pull models on

heterogeneous sources. 3) Due to the scale of the system, all the information of the continuous queries and temporary results cannot be held in memory. A caching mechanism is used to obtain good performance with limited amounts of memory.

Some other projects on data stream processing include StatStream [57],Gigascope [11], TelegraphCQ [22] and so on.

The Grid is developing fast and has been applied in many areas, e.g. data intensive scientific and engineering application workflows, which are based on seamless interactions and coupling between geographically distributed application components. It is common that streaming applications consist of many components, running simultaneously in distributed Grid nodes, processing corresponding stages of the whole workflow. A typical example, proposed in [27], is a fusion simulation consisting of coupled codes running simultaneously on separate HPC resources at supercomputing centers, and interacting at runtime with additional services for interactive data monitoring, online data analysis and visualization, data archiving, and collaboration. The support for asynchronous, high-throughput low-latency data streaming between interacting components is the most important requirement of this type of applications. In the case of the fusion codes, what is required is continuous data streaming from the HPC machine to ancillary data analysis and storage machines. In data streaming applications, data volumes are usually high and data rates are variant, which introduces many challenges, especially in large-scale and highly dynamic environments, Grids for instance, with shared computing and communication resources, resource heterogeneity in terms of capability, capacity and costs, and where application behavior, needs, and performance are highly variable.

The fundamental requirement for Grid data streaming is to efficiently and robustly stream data from data sources to remote services while satisfying the following constraints: (1) Enabling high throughput, low-latency data transfer to support near real-time access to the data. (2) Minimizing overheads on running applications. (3) Adapting to network conditions to maintain desired QoS. (4) Handling network failures while eliminating loss of data.

3 Grid data streaming techniques

3.1 Data stream transfers

In Grid data streaming environments, it is inevitable to transfer large amount of data from data sources, such as simulation programs, running telescopes and other instruments, to remote applications, for further analysis, processing, visualization and so on. Sometimes data transfers are concurrent with the running programs, which generate real-time data, and it should introduce the minimum effect on programs themselves. Some Grid tools can be utilized to implement data transfers.

The Globus Toolkit [40], a widely used grid middleware, provides a number of components for supporting Grid data streaming, and GridFTP [18] and RFT [6] are the two popular tools for data transfers.

GridFTP extends the standard FTP protocol to allow data to be transferred efficiently among remote sites. GridFTP can adjust the TCP buffer and window sizes automatically to

gain the optimal transfer performance. It is secure adopting GSI and kerberos mechanism, and it supports parallel, striped, partial and third-party-controlled data transmissions. However, GridFTP has a shortcoming, i.e., when a client fails, it does not know where to restart the transfer because all the transfer information is stored in the memory, which requires a manual restart for data transfer. To overcome this, the Reliable File Transfer (RFT) service, a non-user based one, was developed. This service is built on top of the GridFTP libraries and stores the transfer requests in a database rather than in memory. Clients are only required to submit a transfer request to the service and do not need to stay active because data transfer is managed by RFT on behalf of the user. When en error is detected, RFT restarts the transfer from the last check point. The transfer status can be retrieved at anytime and RFT can also notify users when a requested transfer is complete.

DiskRouter [60], started as a project at University of Wisconsin-Madison, is aimed to look at network buffering to aid in wide-area data movements. In its present form, it is a piece of software that can be used to dynamically construct an application-level overlay network. It uses hierarchical buffering using memory and disks at the overlay nodes to help in wide-area data movements. DiskRouter provides rich features, such as application-level multicast, running computation on data streams and using higher-level knowledge for data-movements.

3.2 Data stream processing

Data streams are a prevalent and growing source of timely data, and because sequence is indefinite, requests are long running, continuously executing. In a continuous query model, the queries are deployed onto computational nodes in the Grid and execute continuously, over the streaming data.

A group in College of Computing, Georgia Institute of Technology created a middleware solution, called dQUOB [2], short for *dynamic QUery OBject* system, to cope with the challenges of Grid data streaming applications, especially the data flows created when clients request data from a few sources and/or by the delivery of large data sets to clients. Such applications include video-on-demand (VOD), access grid technology to support teleconference for cooperative research, distributed scientific laboratories where the scientists cooperate synchronously via meaningful display of data and computational models, and the large scale data streams that result from digital systems.

The dQUOB system enables users to create queries, perhaps associated with user-defined computations, for precisely the data they aim to use, which will filter data and/or transform it, to put data into the form in which it is most useful to end users. The dQUOB system satisfies clients in need of specific information from high volume data streams, by providing a compiler and run-time environment to create computational entities called quoblets [12] that are insert into high-volume data streams at arbitrary points, including data providers, intermediate machines, and data consumers. Its intent is to reduce end-to-end latency by distributing filtering and processing actions as per resource availability and application needs.

The dQUOB system is able to capture characteristics about the data stream and adapt rules and its management of rule at runtime to detectable patterns of change; to allow specification of complex queries over any number of event types arriving from distributed data source. This system adopts integrated adaptability policy based on database query optimization techniques, conceptualizing a data stream as a set of relational database tables,

for the relational database data model has the significant advantage of presenting opportunities for efficient re-optimizations of queries and sets of queries.

A high-performance stream-oriented distributed database manager and query processor is described in [31], which allows efficient execution of database queries to streamed numerical data from scientific applications. The Grid Stream Database Manager (GSDM) is developed to attain high performance by utilizing many object-relational main-memory database engines distributed in the Grid.

Some problems such as how to distribute the work among the database nodes [13], what type of parallelism would be adopted for scientific data streams with User Defined Functions (UDFs) applied, and how to coordinate the operations from a single pipeline running on different nodes or even clusters, must be addressed, and some optimization and scheduling algorithms have to be developed in the design of the GSDM system, because of the dynamic nature of Grid environment.

In Grid data streaming applications, the volume of data to transfer and process is too large for a single main-memory, so data distribution among clusters of main-memories is desirable. On the other hand, the Grid data streaming application requires very high performance of insert, delete, and data processing operations, which will exceed the ability of single computers or even the clusters. In this case, Computational Grids are more appropriate for GSDM than a regular parallel computer because of dynamics and scalability. Computational Grids are a natural extension of parallel computers, for it can aggregate greater processing power, but more importantly, it is good at scaling in dynamic resource allocation and incorporation of new nodes when necessary and free them when they are not needed any more. The dynamics can also be important characteristics in an environment where different numbers of data streams are used over time or the source streams have varying incoming rates. Grid, as a new Cyber-infrastructure, paves the way for the efficient utilization of computing resources which are distributed and can be aggregated as a whole, to provide large amount of computing capacity for the data streaming applications.

A new kind of database manager, the so-called Grid Data Manager (GDM), is developed in [32], whose target application area is space physics, in particular the LOFAR/LOIS project [37], aiming to develop a distributed software space telescope and radar utilizing the Grid. LOFAR/LOIS will produce extremely large amounts of raw data streams, with rates of several gigabits per second, by sensors receiving signals from space, which imposes demands such as high performance and extensibility on GSM.

3.3 Data stream scheduling

It is common to decompose a data streaming process into several components that interact with each other. The components are executed in a form of pipeline, in which preceding ones' outputs are used as inputs of subsequent ones, just like what happens in Linux pipes. Components executed on different computational nodes will take variant time, since nodes are equipped with different CPUs, memories and software. Also workloads on Grid nodes fluctuate from time to time since resources are shared in a grant scope and the competition makes performance diverse. It is important to make a feasible scheduling for the components, i.e. mapping them to appropriate nodes to achieve optimal performance as a

whole. Performance of data streaming applications is evaluated in terms of throughput, rather than minimizing makespan.

Scheduling in Grids has primarily focused on providing support for batch-oriented jobs, and a wide variety of meta-schedulers and resource brokers using Globus Toolkit have been developed by other research projects such as Condor [62], Legion [63], and Nimrod/G [64]. Condor is aimed to develop, implement, deploy and evaluate mechanism and politics to support high throughput computing (HTC) in the Grid, by efficient use of available resources, i.e., taking wasted computation time and putting it to good use. Condor software is divided into two parts, responsible for job management and resource management respectively. Condor-G is the job management part of Condor, and it applies DAGMan for grid workflow management, where applications are specified by a task-graph, directed acyclic graph (DAG). DAGMan is designed for task-graph based batch jobs with data dependencies and hence launches a task only after all its predecessors have finished execution. Legion is developed as an object-based meta-systems software project at the University of Virginia for a system of millions of hosts and trillions of objects tied together with high-speed links, a virtual super computer. Legion provides transparent scheduling, data management, fault tolerance, site autonomy, and a wide range of security options, which are typical characteristics of the Grid. Nimrod/G is a grid-enabled resource management and scheduling system based on Globus toolkit services and adopts a modular and component-based architecture, with the feature like extensibility, portability, easy development and interoperability of independently developed components. It focuses on the management and scheduling of computations over Grid with particular emphasis on developing scheduling schemes based on the concept of computational economy.

However, in a data streaming application, different from other types of applications, each stage of the processing is concurrently working on a snapshot of the continuous stream data. A simple stream scheduler on top of Condor called E-Condor is demonstrated in [59], which uses Condor to obtain the resources necessary to launch the individual stages of a streaming application. Before launching, E-Condor takes care of setting up all the necessary coupling between the stages commensurate with the data flow graph of the application.

The streaming application in the scheduling problem here is represented by a DAG, in which each node represents a continuously running application stage and the edges denote the direction of data flow. The scheduling system is designed to allocate resources, including computing resources and storage resources to the stages of the application so as to meet the quality of service requirements such as latency and throughout. At some level this resembles multiprocessor scheduling where a task-graph is used as an ordering mechanism to show the dependencies among the individual tasks of a parallel computation. In multiprocessor scheduling, these dependencies are respected and exploited, and the goal is to maximize the utilization of the computational resources and reduce the completion time of the application. Often some heuristic algorithms are applied since the scheduling problem is known to be NP-Complete. On the other hand, the coarse-grain graph of streaming application is a representation of the processing that is carried out on the data during its passage through the pipeline of stages. In the steady state, all the stages of the pipeline are working on a different snapshot of the stream data. So the scheduling of such streaming applications is not an ordering issue; rather, it is a matter of mapping the different stages of the pipeline to the available resources of each stage to optimize the latency and throughput metrics of the entire pipeline.

One important aspect in the usage of distributed and Grid systems is the scheduling of data intensive jobs with respect to the available input data. While scheduling the data streaming applications one obvious principle to remember is the importance of keeping the data flow from sources to destinations alive. In real world data streaming applications contain many steps in a pipeline form, where any kind of interruption at some step of the flow will disrupt the entire process and possibly cause major breakdown because in real-time systems the data are likely to be streamed continuously. For this reason it is wise to break down the data processing applications into as many small components as possible and wrap them with standard interfaces, for example to be wrapped as services, which will allow them to be accessed and controlled easily. This approach helps creating robust real-time data flows because it allows distribution of the processing components and in turn integrating failsafe measures. For instance backup services could be used to replace any failed services thus allowing keeping the flow alive.

3.4 Workflow management

Efficient and robust data streaming applications are frequently composed of several components, each often designed and tuned by a different researcher at geographically distributed computational nodes, keeping seamless runtime interactions, coupling and data relies. A key QoS requirement of these applications is the support for high throughput low-latency robust data streaming between the corresponding distributed components.

Recently, scientific work flows have been used to combine individual applications into large-scale analysis by defining the interactions between the components and the data they rely on. For example, a typical Grid-based fusion simulation workflow [49] consists of coupled simulation codes running simultaneously on separate HPC resources at supercomputing centers, and must interact at runtime with services for interactive data monitoring, online data analysis and visualization, data archiving, and collaboration that also run simultaneously on remote sites.

The scale of data streaming applications and thus of workflows often necessitates that substantial computational, storage and data resources be used to generate the required results. Cyberinfrastructure projects such as TeraGrid [65] and the Open Science Grid (OSG) [66] can provide an execution platform for workflows, but these require a significant amount of end users' expertise to be able to make efficient use of them, so they are not suitable for end users.

So user-friendly tools for workflow management are desirable, and some organizations concerning with these issues have developed some workflow management systems (WFMS). The Workflow Management Coalition (WfMC) [67], founded in 1993, is a global organization of adopters, developers, consultants, analysts, as well as universities and research groups engaged in workflow and BPM. As the unique standards organization concentrating purely on workflow management, the WfMC creates and contributes to process related standards, educates the market on related issues. Some WFMSes have achieved much, but the Grid data streaming applications have their special characteristics that make the common WFMSes not feasible.

Pegasus [5], standing for Planning for Execution in Grids, is a workflow mapping engine developed and used as part of several projects in physics, astronomy, gravitational-wave

science, earthquake science, neuroscience, and others. Pegasus relies on Condor DAGMan workflow engine to bridge the scientific domain and the execution environment by automatically mapping the high-level workflow descriptions onto distributed resources such as TeraGrid, Open Science Grid, and others, and maintain the dependencies between them. Pegasus hides the complexity of Grid resources and allows scientists to construct work flows in abstract terms, which will be translated and mapped to the underlying Cyberinfrastructure. Pegasus is used day-to-day to map complex, large-scale scientific work flows with thousands of tasks processing terabytes of data onto the Grid. As part of the mapping, Pegasus automatically manages data generated during workflow execution by staging them out to user-specified locations, by registering them in data catalogs, and by capturing their provenance information, which is rather desirable in Grid data streaming applications.

Because of the nature of the data streams, especially that of high volumes, it is possible that typical workflow mapping technologies produce work flows that are unable to execute due to the lack of disk space necessary for the successful execution [68]. So a on-demand and in-time cleanup procedure is indispensable, to remove obsolete data which have been processed and are not needed any more to save space for more subsequent data. Currently Pegasus automatically generates a cleanup workflow to delete all data staged-in, and data products generated on the Compute Element after a workflow has finished and the analysis results have been staged out to a user-specified location, i.e., this is a static cleaning up procedure, which introduces significant overhead as the data processing for a single run may require a week of wall time and still occupies big storage capacities. On-demand and in-time cleanup can substantially reduce the storage requirements on the Compute Element during the data analysis.

4 Grid data streaming applications

Grid data streaming has attracted wide interest in the community of scientists and engineers, and it has been widely used, as demonstrated in the following.

4.1 Data stream mining

In general, data mining (sometimes called data or knowledge discovery) is the process of analyzing data from different perspectives and summarizing it into useful information [69]. Sharing and collaboration of Grid resources, such as computation, storage, and relative software are dramatically increasing the accuracy of analysis while driving down the cost.

Stream data mining tasks include performing multi-dimensional statistical analysis, clustering and classifying data streams, finding frequent patterns [70] and sequential patterns, setting alarms for monitoring and so on. Compared to mining from a static transaction data set, the streaming case has far more information to track and far greater complexity to manage. Characteristics of stream data impose some challenges in stream data analysis and processing, so some special algorithms are in bad need, such as a tilted time window to aggregate data at different points in time, a scalable multi-dimensional stream data cube that can aggregate a model of stream data efficiently without accessing the raw data.

Mining alarming incidents in data streams, MAIDS [71], whose prototype was completed in April 2004, aims to perform a systematic investigation of stream data mining principles and algorithms, develop effective, efficient, and scalable methods for mining the dynamics of data streams, and implement a system prototype for online multi-dimensional stream data mining applications. By developing and implementing some new and existing algorithms, it tries to discover changes, trends and evolution characteristics in data streams; construct clusters and classification models from data streams; and explore frequent patterns and similarities among data streams. Its applications include network intrusion detection, telecommunication data flow analysis, credit card fraud prevention, Web click streams analysis, financial data trend prediction, and other applications. MAIDS is distinguished in the terms of its features, i.e. general purpose tools for data stream analysis, ability to process high-rate and multi-dimensional data, a flexible tilted time window framework, facilitating multi-dimensional analysis using a stream cube architecture, multiple data mining functions, user-friendly interface, automatic analysis and on-demand analysis, easy to set alarms for monitoring.

MAIDS consists of several components, including a statistics query engine, which answers user queries on data statistics, such as, count, max, min, average, regression, etc; a stream data classifier, which builds models to make predictions and sets alarm to monitor events; and a stream pattern finder, which finds frequent patterns with multiple time granularities and mines evolution and dramatic changes of frequent patterns.

MAIDS mainly deals with Sequential Patter Matching and Hidden Network Mining. The former tries to discover frequent subsequences as patterns in a sequence database, whose applications include the analysis of customer purchase patterns, the analysis of DNA, and the analysis of sequenced or time-related processes such as those found in scientific experiments, natural disasters, and disease treatments, keeping a huge impact in homeland security, network intrusion detection, and bioinformatics; the latter is focusing on discovering the complex, hidden social networks, such as telecommunications, web activity, co-authorship of texts, attending a party or a meeting, participation in sports, or even contracting a disease, behind real world entities, and it has a strong impact in marketing, crime and national security investigations, social network analysis and other fields.

4.2 Large-scale simulation

Large scale simulations are increasingly important in many fields of science. A major fusion plasma simulation, the Gyrokinetic Toroidal Code (GTC) [8][9][27], examines the highly complex, non-linear dynamics of plasma turbulence using direct numerical simulations, and currently generates about 1TB/week of simulation results data during production use. A system which efficiently and automatically transfers chunks of data from the simulation to a local analysis cluster during execution has been developed. By overlapping the simulation with the data transfer and the analysis, scientists can analyze their results as they are being produced.

The scientists used to place the generated computational data on the supercomputing sites and transfer the data manually after the execution is finished, or, to execute remote visualization and post-processing of the data. Owe to the high volumes of data, both approaches are not suitable, for example, remote visualization gives birth to issues of latency

and network quality of service, which forces the scientists to put forward new ways for data transfer and remote visualization. And the quality of service requirements can be summarized as low latency, high throughput, high reliability and minimum impact on the running simulation itself.

Grid computing has made a major impact in the field of collaboration and resource sharing [1]. Especially, data Grids are generating a tremendous amount of excitement in Grid computing [18]. More and more visualization scientists are using the Grid to develop collaborative interactive visualization techniques [50][51][52][53]. In GTC, it is preferable to transfer data from a supercomputer running the main simulation on N processors, to M processors (typically M<N) on a visualization/analysis cluster local to the scientist, where it is convenient to perform the post-processing of the data, which requires much less computing power than the main simulation. The processors can be viewed as Grid nodes with well-designed scheduling and management policy and feasible distribution of data, so the post-processing can be parallel so as to attain high performance to keep pace with the main simulation.

To keep the overhead in simulations minimum while utilizing Grid resources to the maximum, a scenario of the threaded buffer for data transfer has implemented, which uses simple APIs to activate the transfer. To make this data transfer efficient with the added advantage of global scheduling, optimization of data movement, storage and computation, Logistical Networking (LN) [54] is introduced which is built on the Internet Backplane Protocol (IBP). Except network bandwidth, LN allows for a flexible sharing and utilization of storage resources, so it is natural to construct a pipeline [28] for data flow with various depots (storage) locations containing the data in various stages of transformation. The existence of pervasive depots aids in the creation of a reliable data pipeline, allowing simulations to transparently store data to adjacent depots in case of network failures at the receiving end or buffer overflows at the sending end to guarantee that no data is lost. A so-called pull/fetch mechanism makes sure that part of data can be transferred to its pre-determined destination, and enables us to utilize the network bandwidth maximally and affect the simulation's performance minimally. Some methods have been proposed to utilize the bandwidth efficiently [50].

Alternatively, a threaded parallel data streaming approach has been developed in [8], using Globus to transfer multi-terabyte simulation data from a remote supercomputer to the scientist's home analysis/visualization cluster, concurrent with the simulation, with negligible overhead. It is proved that this concurrent data transfer approach is more favorable compared with writing to local disk and then transferring this data to be post-processed after the simulation has been finished, for the scientists can get the preliminary results much sooner, which is fundamentally important for them to judge whether or not the simulation is going in the right way and carry out on-line control on it.

Obviously, we can benefit from the seamless combination of simulation, data transfer and post-processing, i.e. in a form of pipelines. GridFTP is perhaps the most popular data transfer tool in Grid environments, and it provides many useful application program interfaces (APIs) which can be adapted and extended to integrate post-processing. This has been demonstrated in experiments included in [8].

4.3 Multiple data stream processing

The field of e-Science [25] currently faces the challenge to analyze huge volumes of scientific data which are distributed in various sciences and communities, in which scientists share scientific interests, data, and research results. These issues can be addressed by processing large data volumes on-the-fly in the form of data streams and by combining multiple data sources and making the results available in a network.

Processing of multiple data streams in Grid-based peer-to-peer (P2P) networks is described in [20]. Spatial matching, a current issue in astrophysics as a real-life e-Science scenario, is introduced to show how a data stream management system (DSMS) can help in efficiently performing associated tasks. Actually, spatial matching is a job of information fusion across multiple data sources, where transmitting all the necessary data from the data sources to the data sink for processing (data shipping) is problematic and in many cases will not be feasible any more in the near future due to the large and increasing data volumes. The promising solutions are dispersing executing operators that reduce data volumes at or near the data sources (query shipping) or distributing query processing operators in a network (in-network query processing). In-network query processing, as employed in the StreamGlobe [72][73] system, can also be combined with parallel processing and pipelined processing of data streams, which enables further improvements of performance and response time in e-Science workflows.

The Laser Interferometer Gravitational-Wave Observatory (LIGO) [74] is a facility dedicated to the detection of cosmic gravitational waves and the harnessing of these waves for scientific research. It consists of two widely separated installations within the United States — one in Hanford Washington and the other in Livingston, Louisiana — operated in unison as a single observatory. Some LIGO data analysis programs receive different types of data streams and carries out signal processing or statistical analysis.

4.4 Sensor grid data streaming

Recent advancements in sensor technologies such as micro-circuitry, nano-technology and low-power electronics have allowed sensors to be deployed in a wide variety of environments [15][16][26]. In the near future thousands of sensor nodes will be deployed either individually or as part of sensor networks in a large variety of application domains, such as environmental monitoring, air pollution and water quality measurements, detection of the seismic events, and understanding the long-term motions of the Earth crust. Extensive use of sensing devices and deployment of the networks of sensors that can communicate with each other to achieve a larger sensing task will fundamentally change information gathering and processing [54].

It is safe to say, judging from the development of a common Grid infrastructure for Geographic Information Systems, sensor grid can produce very large amounts of observational data that potentially may help us to get a better knowledge of the environment we live in, and we can expect that the real-time sensor measurements will become the dominant type of timely data sources, which may be more important than traditional systems. For instance Southern California Integrated GPS Network (SCIGN) [56] has deployed 250 continuously-operating GPS stations in Southern California whereas several hundred non-real

time stations are operating elsewhere in the United States [41]. The GPS Earth Observation Network System or GEONET in Japan consists of an array of 1200 GPS stations that covers the entire country with an average spacing of about 20km [57]. These networks are capable of producing terabytes of measurements per year which constitutes great challenges for the processing and storage capacities of the current systems.

Appropriate mechanisms are in great need to manage, organize and transform the real-time sensor Grid data streams that are produced by the devices, and to deliver this data to applications to draw conclusions and knowledge. So some processing components, called filters in Sensor Grids, are designed to accept transforms and presents messages. Depending on the tasks they are designed to accomplish, they may be small applications such as format converters or very complex applications like simulation software, may be expected to run in real-time by immediately processing and presenting the results or in near-real time by accumulating certain amount of data and executing the processing afterwards.

In Sensor Grids the filters are deployed as Web Services and in most cases run in a workflow as part of complex processes. Each filter is designed to execute a particular task, which is to say that it accepts and produces certain type of messages, so there are dependencies between the filters to act coordinately in a workflow. These dependencies must be respected in constructing a proper workflow, so it is wise to define these dependencies in a filter specific metadata document for checking if these dependencies are satisfied at the time of the deployment. In this way users will see which other filters must be deployed as well.

4.5 Audio and video stream services and management

Generally speaking, there are a lot of similarities between multimedia streams and other data streams such as sensor data mentioned in the foregoing sub-section. For example, all streaming data require significant Quality of Service (QoS) constraints and dynamic filtering. Raw multimedia bit-streams will occupy too high bandwidth, so complicated codes must be used to compress the streams and transmit them over the Internet. Further, multimedia streams are typically used collaboratively and multicast to multiple clients.

Global multimedia collaboration system [75][76], Global-MMCS, as a service-oriented multimedia collaboration system, mainly processes multimedia streams: video, audio, whiteboards and so on, which supports a diversity of collaboration clients. The basic services include those to support audio-video collaboration, such as reliable delivery, multicasting and replay.

5 Research chanllenges and our proposal

Grid data streaming techniques and applications are summarized in previous sections. In this section, we identify some existing challenges in a grid data streaming environment and propose our designs and implementation with some initial experimental results.

5.1 On-demand data stream scheduling

In the most applications mentioned above, the DSMSs are trying to transfer data from their sources to destinations complying with the principle: the faster the better or as soon as possible. Sometimes these principles work well, especially in the cases of real-time simulations or running instruments. In some circumstances, data overflow may cause wastes of storage and bandwidth.

There may be too many backlogs in the system waiting for processing, which consume too many storage resources. The subsequent coming data may be lost due to storage shortage, which is not acceptable. Obviously, on-demand data transfers are more preferable, which means that data should be transferred when and only when it is needed, i.e., the data transfers are controlled, not spontaneous. In this scenario, some factors must be taken into account, including data processing speed, network bandwidth and available storage capacity.

We come up with a data streaming scheduling scenario which resembles the repertory strategy in Operation Research (OR). We set the upper limits and lower limits for different types of data anticipating the processing and control data transfers according to the limits: when the amount of some type of data in the storage is under the lower limit, the data transfers are started, until it reaches the upper limit. Our principle is to store appropriate amounts of data, meeting processing requirements with minimum storage space and bandwidth requirement. The lower limit is used to guarantee that data processing can survive network collapse when no data can be streamed from sources to local storage for a certain period of time, which improves system robustness and increases CPU resource utilization. The upper limit for each application is used to guarantee that the overall amount of data in local storage does not exceed available storage space.

There is always more than one data type to be scheduled, so it is not straightforward to set appropriate lower and upper limits for them manually, especially in a dynamic Grid environment. Some heuristic method must be applied, and our choice is the genetic algorithm (GA) [77][78], which allows us to find the satisfying, not necessarily optimal, solutions without precise mathematical models in an evolutionary environment. Preliminary experiments show good performance of our on-demand data transfers in both scalability and adaptability.

5.2 Performance measurements and prediction

As mentioned above, at least three factors, i.e., processing speeds, network bandwidth and available storage, influence the lower and upper limits in our on-demand data streaming scenario. A Grid environment is dynamic and changing where none of the three factors are constant, which makes scheduling complex. Obviously, our scheduling requires measurement prediction ability to performance of these three factors.

It is hard to find precise analytical models for performance prediction on these aspects, but historical information can give us some hints, that is to say, we can infer the performance for a short period of future from the historical data. There have been many methods for this purpose, for example, Auto-regressive (AR), Moving Average (MA), Auto-regressive Moving Average (ARMA), Auto-regressive Integrated Moving Average (ARIMA), Auto-regressive Fractionally Integrated Moving Average (ARFIMA), LAST, MEAN. We adopt the

fractal method[79][80], which proves to be effective. A detailed introduction to the method is out of scope of this article.

We sample the data from CPUs, network and so on at a frequency of 1Hz, and put these data into sequence. A fractal method is applied, which produces the prediction values. The predictions coincide with the actual ones measured in real-time to a great extent, and the prediction can reflect the trend of actual performance. Our prediction algorithm is light-weighted and its overhead can be neglected. For the dynamic nature of Grid, the prediction results are only effective for a relatively short time, so the prediction must be also running constantly, taking the most recent historical data into accounts and giving them higher weight since these reflect the current status more than stale records.

5.3 System implementation

As shown in Figure 1, implementation of the data streaming environment is described in this section, including individual processors for performance sensing, prediction, scheduling and execution of data transfers and cleanups.

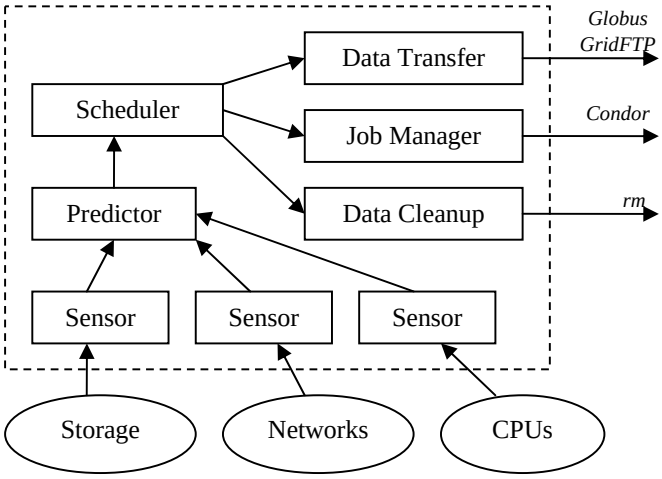


Figure 1. The Data Streaming Environment

As mentioned above, since the Grid is a dynamic environment, real-time performance information is required. Some sensors are developed to collect such information, which can be used as inputs of the performance predictor. The predictor applies some prediction algorithms to forecast performance for a period of time in the future. Prediction results are inputs of the scheduler. The scheduler collects performance prediction results (e.g. data transferring and processing speeds) and decides when to transfer data, launch jobs and clean up processed data. Actual executions call external functions provided by Globus and Condor, as shown in Figure 1.

Our optimal goal is to make full use of computational resources, requiring guarantee of data provision, and minimize bandwidth and storage consumptions.

5.4 Experimental results

A prototype data streaming environment is implemented using a Condor pool to manage over 10 workstations in the laboratory and providing computational and storage resources. Data to be processed are available in remote sites rather than machines in the Condor pool, and Globus GridFTP is adopted for data streaming.

Three repertory strategies are used for performance evaluation of storage scalability, which are illustrated in Figure 2. In the mode of continuous streaming without cleanup, data transfers are not under any control, i.e. data are transferred constantly, and no cleanup procedure is involved, so data accumulates rapidly. After cleanup process is added, while the required storage space scales better, data still accumulates over time. This is because data transfers are faster than processing. Although processed data can be removed immediately, data waiting to be processed still accumulates and required storage space increases. In on-demand streaming with cleanup proposed in this work, data transfers are controlled using lower and upper limits, so data transfers would stop at upper limits and data overflow would not happen.

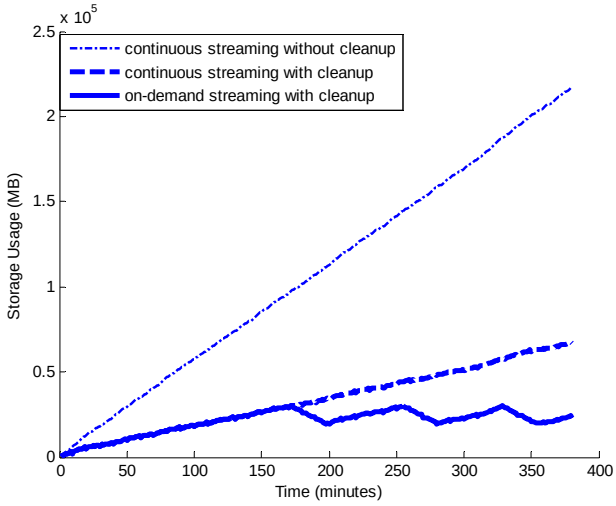
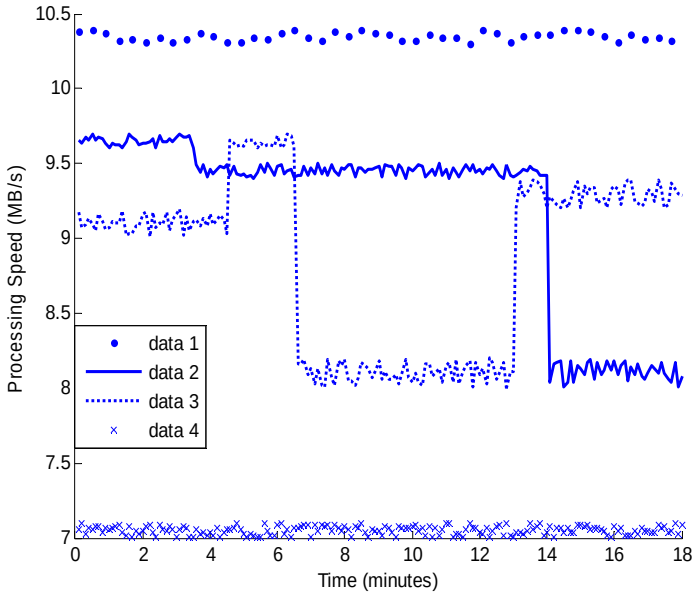


Figure 2. Performance Evaluation on Storage Scalability using 3 Repertory Strategies

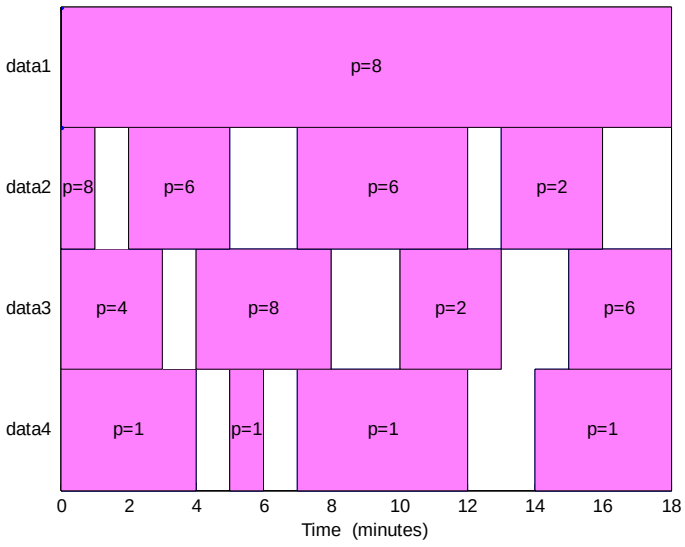
From experimental results given in Figure 2 we can infer that our data streaming and scheduling mechanism should scale well with applications requiring high volumes of data and long-term runs.

Figure 3 shows a scheduling scheme for 4 processing programs. Since the *data1* application has a very high data processing speed, the GridFTP parallelism is set to the highest 8 in order to meet the requirement (further increasing of the GridFTP parallelism would not accelerate the data transferring speed). Even data has been streamed as fast as possible, it is obvious that data processing still has to be idle and wait for available data occasionally, as shown in Figure 3a. On the other hand, *data4* is slow so *p=1* can always satisfy the application, even with some intervals when no data transfers happen, as shown in Figure 3b. As shown in Figure 3a, *data2* and *data3*'s processing speeds change dramatically over time. The data streaming environment has to better adapt to dynamic requirements by

changing *p* on-the-fly from 2 to 8. Also as shown in Figure 3b, data transfers can start (when lower limits in repertory policies are reached) and stop (when upper limits are reached) automatically, meeting application requirements as well as avoiding data overflows.



(a)



(b)

Figure 3. Performance Evaluation on Adaptability of Data Streaming to Dynamically Changing Data Processing Speeds

6 Conclusions

Grid data streaming applications are popular in many fields, such as high energy physics, astronomical observations, large scale simulations, multimedia collaboration, sensor network, etc. A survey of grid data streaming techniques and applications is provided in this work.

A prototype of a data streaming environment is implemented, together with Condor for CPU allocation and Globus GridFTP for data transfers. Some repertory strategies are defined, performance measurement and prediction methods are proposed, and scheduling algorithms are implemented. Experimental results provide reasonable performance evaluation on both scalability and adaptability of our environment.

In the future, we'd like to make further research on the on-demand data streaming scheduling, performance prediction, fault-tolerate mechanisms and workflow management to achieve higher performance in terms of low latency and high throughput.

References

[1]. I. Foster, and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, San Francisco, 1998.

[2]. B. Plale, and K. Schwan, "dQUOB: Managing Large Data Flows Using Dynamic Embedded Queries", in *Proc. IEEE Int. High Performance Distributed Computing*, 263-270, 2000.

[3]. A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke, "The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Data Sets", *Journal of Network and Computer Applications*, 23(3): 187-200, 2001.

[4]. A. Chervenak, E. Deelman, I. Foster, L. Guy, W. Hoschek, A. Iamnitchi, C. Kesselman, P. Kunszt, M. Ripeanu, B. Schwartzkopf, H. Stockinger, K. Stockinger, and B. Tierney, "Giggle: a Framework for Constructing Scalable Replica Location Services", in *Proc. IEEE Supercomputing*, Baltimore, 2002.

[5]. E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M. Su, K. Vahi, and M. Livny, "Pegasus: Mapping Scientific Workflows onto the Grid", in *Proc. 2nd European Across Grids Conf.*, Nicosia, Cyprus, 2004.

[6]. R. Madduri, C. Hood, and W. Allcock, "Reliable File Transfers in Grid Environments", in *Proc. 27th IEEE Conf. on Local Computer Networks*, 737-738, Tampa, Florida, USA, 2002.

[7]. M. Manohar, A. Chervenak, B. Clifford and C. Kesselman, "A Replica Location Grid Service Implementation", in *Data Area Workshop, Global Grid Forum 10*, 2004.

[8]. V. Bhat, S. Klasky, S. Atchley, M. Beck, D. McCune and M. Parashar, "High Performance Threaded Data Streaming for Large Scale Simulations", in *Proc. 5th IEEE/ACM Int. Workshop on Grid Computing*, 243-250, Pittsburgh, USA, 2004.

[9]. S. Klasky, M. Beck, V. Bhat, E. Feibush, B. Ludcher, M. Parashar, A. Shoshani, D. Silver and M. Vouk, "Data Management on the Fusion Computational Pipeline", *Journal of Physics: Conf. Series 16*, 510-520, 2005.

[10]. B. Shivanath and W. Jennifer, "Continuous Queries over Data Streams", *SIGMOD Record*, 30(3):109-

120, 2001.

[11]. C. Cranoe, T. Johnson, V. Shkapenyuk, and O. Spatscheck. "Gigascope: a Stream Database for Network Applications", in *Proc. Int. Conf. on Management of Data*, 647-651, 2003.

[12]. B. Plale and K. Schwan, "Dynamic Querying of Streaming Data with the dQUOB System", *IEEE Transactions in Parallel and Distributed Systems*, 14(4): 422-432, 2003.

[13]. N. Sivaramakris, T. Kurc, U. Catalyurek, and J. Saltz, "Database Support for Data-Driven Scientific Applications in the Grid", *Grid Parallel Processing Letters*, 13(2): 245-271, 2003.

[14]. B. Plale, "Architecture for Accessing Data Streams on the Grid", in *Proc. 2nd European Across Grids Conf.*, 2004.

[15]. G. Aydin, Z. Qi, M. E. Pierce, Y. Bock, and G. C. Fox., "Building a Sensor Grid for Real Time Global Positioning System Data", in *Proc. Workshop on Principles of Pervasive Information Systems Design Sunday*, 2007, in conjunction with *Pervasive 2007* Toronto, Ontario, Canada.

[16]. L.F. Akyildiz, W.L. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks", *IEEE Communications Magazine*, 40(8): 102-114, 2002.

[17]. M. Wan, A. Rajasekar, R. Moore, and P. Andrew, "A Simple Mass Storage System for the SRB Data Grid", in *Proc. 20th IEEE Symp. on Mass Storage Systems and 11th Goddard Conf. on Mass Storage Systems and Technologies*, 20-25, San Diego, 2003.

[18]. W. Allcock, J. Bester, and J. Bresnahan et al, "Data Management and Transfer in High Performance Computational Grid Environments", *Parallel Computing Journal*, 28 (5): 749-771, 2002.

[19]. O. Tatebe, Y. Morita, S. Matsuoka, H. Sato, S. Sekiquchi, Y. Watase, M. Imori, and T. Kobayashi, "Grid Data Farm for Petascale Data Intensive Computing", *Electrotechnical Laboratory, Technical Report, TR-2001-4*, http://datafarm.apgrid.org

[20]. R. Kuntschke, T. Scholl, S. Huber, A. Kemper, A. Reiser, H. Adorf, G. Lemson, and W. Voges, "Grid-based Data Stream Processing in e-Science", in *Proc. 2nd IEEE Int. Conf. on e-Science and Grid Computing*, Amsterdam, The Netherlands, 2006.

[21]. A. Arasu, B. Babcock, S. Babu, M. Datar, K. Ito, R. Motwani, I. Nishizawa, U. Srivastava, D. Thomas, R. Varma, and J. Widom, "STREAM: The Stanford Stream Data Manager", *IEEE Data Engineering Bulletin*, 26(1): 19–26, 2003.

[22]. S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J. M. Hellerstein, W. Hong, S. Krishnamurthy, S. Madden, V. Raman, F. Reiss, and M. A. Shah, "TelegraphCQ: Continuous Dataflow Processing for an Uncertain World", in *Proc. Conf. on Innovative Data Systems Research*, Asilomar, CA, USA, 2003.

[23]. J. Chen, D. J. DeWitt, F. Tian, and Y. Wang, "NiagaraCQ: a Scalable Continuous Query System for Internet Databases", in *Proc. the ACM SIGMOD Int. Conf. on Management of Data*, 379–390, Dallas, TX, USA, 2000.

[24]. L. Chen, K. Reddy, and G. Agrawal, "GATES: a Grid-Based Middleware for Processing Distributed Data Streams", in *Proc. IEEE Int. Symp. on High-Performance Distributed Computing*, 192–201, Honolulu, HI, USA, June 2004.

[25]. S. Huber, "Grid-based Processing of Multiple Data Streams in E-Science", *Master's thesis, Technische*

Universität München, 2005.

[26]. Y. Yao and J. Gehrke, "Query Processing for Sensor Networks", in *Proc. Conf. on Innovative Data Systems Research*, Asilomar, CA, USA, 2003.

[27]. S. Klasky, S. Ethier, and Z. Lin et al., "Grid-Based Parallel Data Streaming Implemented for the Gyrokinetic Toroidal Code", in *Proc. ACM/IEEE SC2003 Conf.*, Phoenix, Arizona, USA, 2003.

[28]. T. Kosar, G. Kola, and M. Livny, "Building Data Pipelines for High Performance Bulk Data Transfers in a Heterogeneous Grid Environment", *Technical Report CS-TR-2003-1487, Computer Sciences Department, University of Wisconsin- Madison*, 2003.

[29]. A. Sim, J. Gu, A. Shoshani, and V. Natarajan, "DataMover: Robust Terabyte-Scale Multi-File Replication over Wide-Area Networks", in *Proc. 16th Int. Conf. on Scientific and Statistical Database Management*, Santorini Island, Greece, 2004.

[30]. A. L. Chervenak, N. Palavalli, S. Bharathi, C. Kesselman, R. Schwartzkopf, "Performance and Scalability of a Replica Location Service", in *Proc. 13th Int. IEEE Symp. on High Performance Distributed Computing*, 182-191, 2004.

[31]. M. G. Koparanova, and T. Risch, "High-Performance Grid Stream Database Manager for Scientific Data", in *Proc* 1st *European Across Grids Conf.*, 86-92, Universidad de Santiago de Compostela, Spain, 2003.

[32]. M. G. Koparanova, and T. Risch, "High-performance Stream-oriented GRID Database Manager for Scientific Data", in *Proc. 1st European Across Grids Conf.*, Universidad de Santiago de Compostela, Spain, 2003.

[33]. P. Seshadri, M. Livny, and R. Ramakrishnan, "SEQ: a Model for Sequence Databases", in *Proc. 11th Int. Conf. on Data Engineering*, 232-239, Taipei, Taiwan, 1995.

[34]. B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, "Models and Issues in Data Stream Systems", in *Proc. 21st ACM Symp. on Principles of Database Systems*, 1-16, 2002.

[35]. M. Cherniack, H. Balakrishnan, M. Balazinska, D. Carney, U. Cetintemel, Y. Xing, and S. Zdonik, "Scalable Distributed Stream Processing", in *Proc. 1st Biennial Conf. on Innovative Data System Research*, 2003.

[36]. D. Carney, U. Cetinterne, M. Cherniack, et. al., "Monitoring Streams: a New Class of Data Management Applications", in *Proc. Int. Conf. on Very Large Data Bases*, 215-225, 2002.

[37]. B. Thide (ed.): *First LOFAR/LOIS Workshop*, Sweden, 2001.

[38]. P. Bonnet, J. Gehrke, and P. Seshadri, "Towards Sensor Database Systems", in *Proc. 2nd Int. Conf. on Mobile Data Management*, 3-14, Hong Kong, 2001.

[39]. B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, "Models and Issues in Data Stream Systems", in *Proc. 21st ACM Symp. on Principles of Database Systems*, 1-16, 2002.

[40]. I. Foster, and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit", *Int. J. Supercomputer Applications*, 11(2): 115-128, 1997.

[41]. G. Fox, G. Aydin, H. Gadgil, S. Pallickara, M. Pierce, and W. Wu, "Management of Real-Time Streaming Data Grid Services", *Invited Talk at 4th Int. Conf. on Grid and Cooperative Computing*, Beijing, China, 2005.

[42]. A. Arasu, and B Babcock et al, "STREAM： the Stanford Stream Data Manager", *IEEE Data Engineering Bulletin*, 26(1): 19-26, 2003

[43]. R. Avnur, and J. Hellerstein, "Eddies: Continuously Adaptive Query Processing", in *Proc. SIGMOD Int. Conf. on Management of Data*. 261-272, New York, 2000.

[44]. D. Duellmann, W. Hoschek, J. Jean-Martinez, A. Samar, and B. Segal, "Models for Replica Synchronization and Consistency in Data Grid", in *Proc. 10th IEEE Symp. on High Performance and Distributed Computing*, 2001.

[45]. I. Foster, J. Vockler, M. Wilde, and Y. Zhao, "The Virtual Data Grid: a New Model and Architecture for Data Intensive Collaboration", in *Proc. CIDR 1st Biennial Conf. on Innovative Data Systems Research*, 2003.

[46]. A. Chervenak, R. Schuler, C. Kesselman, S. Koranda, and B. Moe, "Wide Area Data Replication for Scientific Collaborations", in *Proc. 6th IEEE/ACM Int. Workshop on Grid Computing*, Seattle, WA, USA, 2005.

[47]. L. Golab, and M. T. Ozsu, "Issues in Data Stream Management", *SIGMOD Record*, 32(2): 5-14, 2003.

[48]. P. Seshadri, M. Livny, and R. Ramakrishnan, "The Design and Implementation of a Sequence Database System", in *Proc. 22nd Conf. on Very Large Data Bases*, 99-110, 1996.

[49]. V. Bhat, M. Parashar, M. Khandekar, N. Kandasamy, and S. Klasky, "A Self-Managing Wide-Area Data Streaming Service using Model-based Online Control", in *Proc. 7th IEEE Int. Conf. on Grid Computing*, 176-183, Barcelona, Spain, 2006.

[50]. W. Bethel, and J. Shalf, "Consuming Network Bandwidth with Visapult", *The Visualization Handbook*. Ed. J. Hansen, Academic Press, 2003.

[51]. M. D. Beynon, T. Kurc, A. Sussman, and J. Saltz, "Design of a Framework for Data-Intensive Wide-Area Applications", in *Proc. 9th Heterogeneous Computing Workshop*, Cancun, Mexico, 2000.

[52]. H. Andrade, M. Beynon, C. Chang, R. Ferreira, U. Catalyurek, T. Kurc, A. Sussman and J. Saltz, "Processing Large Scale Multidimensional Data in Parallel and Distributed Environments", *Parallel Computing, Special Issue on Data Intensive Computing*, 28(5): 827-859, 2002.

[53]. W. Allcock, J. Bester, J. Bresnahan, I. Foster, J. Gawor, J. Insley, J. Link, and M. Papka, "GridMapper: a Tool for Visualizing the Behavior of Large Scale Distributed Systems", in *Proc. 11th IEEE Int. Symp. on High Performance Distributed Computing*, 179-188, 2002.

[54]. M. Beck, and T. Moore, "An End-to-End Approach to Globally Scalable Network Storage", *ACM SIGCOMM 2002*, Micah Beck, Pittsburgh, PA, USA, 2002.

[55]. D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, *Next Century Challenges: Scalable Coordination in Sensor Networks*, 263-270, ACM Press New York, NY, USA. 1999.

[56]. K.W. Hudnut, and Y. Bock, "The Southern California Integrated GPS Network (SCIGN)", in *Proc. Int. Workshop on Seismotectonics at the Subduction Zone, Tsukuba*, 1999.

[57]. A. Yamagiwa, Y. Bock, and J. Genrich, "Real-time Monitoring of Crustal Deformation Using Large GPS Geodetic Networks-Japanese GEONET's Potential as a Natural Hazards Mitigation System", in *American Geophysical Union, Fall Meeting 2004*, abstract SF53A-0724, 2004.

[58]. Y. Zhu, and D. Sasha, "StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time", *Technical Report TR2002-827, CS Dept, New York University*, 2002.

[59]. B. Agarwalla, N. Ahmed, D. Hilley, and U. Ramachandran, "Streamline: a Scheduling Heuristic for Streaming Applications on the Grid", in *Proc. SPIE Multimedia Computing and Networking*, Vol. 6071, 2006.

[60]. G. Kola and M. Livny, "DiskRouter: A Mechanism for High Performance Large Scale Data Transfers", http://www.cs.wisc.edu/condor/diskrouter/, 2004.

[61]. P. Sendín-Rana, N. Otero-Alonso, V.G. de Miguel, and F.J. González-Castaño, "Emulating Access Grid Features at Web Endpoints", in *Proc. 6th Int. Conf. on Grid and Cooperative Computing*, 507-512, Universidad de Vigo, Spain, 2007.

[62]. M. Litzkow, M. Livny, and M. Mutka, "Condor- a Hunter of Idle Workstations", in *Proc. 8th Int. Conf. on Distributed Computing Systems*, 104-111, 1988.

[63]. S. J. Chapin, D. Katramatos, J. Karpovich, and A.S. Grimshaw, "The Legion Resource Management System", *Job Scheduling Strategies for Parallel Processing*, 162-178, Springer Verlag, 1999.

[64]. R. Buyya, D. Abramson, and J. Giddy, "Nimrod/G: an Architecture for a Resource Management and Scheduling System in a Global Computational Grid", in *Proc. Int. Conf. on High Performance Computing in Asia-Pacific Region*, 2000.

[65]. P. R. Woodward, S. E. Anderson, D. H. Porter, and A. Iyer, "Distributed Computing in the SHMOD Framework on the NSF TeraGrid", *Technical Report, CS Dept, University of Minnesota*, 2004.

[66]. R. Pordes for the Open Science Grid Consortium, "The Open Science Grid", in *Proc. Computing in High Energy and Nuclear Physics Conf.*, Interlaken, Switzerland, 2004.

[67]. D. Hollingsworth, *The Workflow Reference Model*, Workflow Management Coalition, 1995.

[68]. A. Ramakrishnan, G. Singh, H. Zhao, E. Deelman, R. Sakellariou, K. Vahi, K. Blackburn, D. Meyers, and M. Samidi, "Scheduling Data-Intensive Workflow onto Storage-Constrained Distributed Resources", in *Proc. 7th IEEE Int. Symp. on Cluster Computing and the Grid*, 401-409, Rio de Janeiro-Brazil, 2007.

[69]. H. Kargupta, R. Bhargava, K. Liu, M. Powers, P. Blair, S. Bushra, J. Dull, K. Sarkar, M. Klein, M. Vasa, and D. Handy, "VEDAS: A Mobile and Distributed Data Stream Mining System for Real-Time Vehicle Monitoring", in *Proc. 4th SIAM Int. Conf. on Data Mining*, Lake Buena Vista, USA, 2004.

[70]. C. Giannella, J. Han, J. Pei, X. Yan, and P. S. Yu, *Mining Frequent Patterns in Data Streams at Multiple Time Granularities*, Cambridge, Mass: MIT Press, 2003.

[71]. Y. D. Cai, D. Clutter, G. Pape, J. Han, M. Welge, L. Auvil, "MAIDS: Mining Alarming Incidents from Data Streams", in *Proc. 23rd ACM SIGMOD*, Paris, France, 2004.

[72]. R. Kuntschke, B. Stegmaier, A. Kemper, and A. Reiser, "StreamGlobe: Processing and Sharing Data Streams in Grid-Based P2P Infrastructures", in *Proc. Int. Conf. on Very Large Data Bases*, 1259–1262, Rondheim, Norway, 2005.

[73]. B. Stegmaier, R. Kuntschke, and A. Kemper, "StreamGlobe: Adaptive Query Processing and Optimization in Streaming P2P Environments", in *Proc. Int. Workshop on Data Management for Sensor Networks*, 88–97, Toronto, Canada, 2004.

[74]. E. Deelman, C. Kesselman, G. Mehta, L. Meshkat, L. Pearlman, K. Blackburn, P. Ehrens, A. Lazzarini, R. Williams, and S. Koranda, "GriPhyN and LIGO, Building a Virtual Data Grid for Gravitational Wave Scientists", in *Proc. 11th IEEE Int. Symp. on High Performance Distributed Computing*, 225-234, 2002.

[75]. G. Fox, W. Wu, A. Uyar, H. Bulut, S. Pallickara, "Global Multimedia Collaboration System", in Proc. *1st Int. Workshop on Middleware for Grid Computing*, Rio de Janeiro, Brazil, 2003.

[76]. G. Fox, W. Wu, A. Uyar, H. Bulut, "A Web Services Framework for Collaboration and Audio/Videoconferencing", in *Proc. Int. Multiconf. in Computer Science and Computer Engineering, Internet Computing*, Las Vegas, USA, 2002.

[77]. J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.

[78]. B. P. Buckles, and F. E. Perry , *Genetic Algorithms*, IEEE Computer Society Press, Los Alamitos, CA, 1992.

[79]. J. Feder, *Fractals*, Plenum, N.Y., 1988.

[80]. B. Mandelbrot, *The Fractal Geometry of Nature*, Freeman, N.Y., 1983.