

A Services Oriented Framework for Next Generation Data Analysis Centers *

H. Wang, A. Ghoting, G. Buehrer, S. Tatikonda, S. Parthasarathy, T. Kurc, and J. Saltz
The Ohio State University
Contact Email: srini@cse.ohio-state.edu

Abstract

Over the past decade, advances in computational and sensor technology have enabled us to dynamically collect vast amounts of data from observations, health screening tests, simulations, and experiments at an ever-increasing pace. Knowledge discovery and data mining is an iterative process concerned with deriving interesting, non-obvious, and useful patterns and models from such large volumes of data. Although inexpensive storage is conducive to maintaining said data, accessing and managing it for knowledge discovery and data mining becomes a performance issue when datasets are large, dynamic, and distributed. In this work, we present our vision of a software framework consisting of middleware services to support interactive data mining over dynamic data at data analysis centers built on top of heterogeneous clusters. The design of a sampling service for dynamic data, together with initial performance results, are also presented.

1. Introduction

Understanding complex processes oftentimes involves the development of a detailed model of their data. This is a *dynamic, data-driven* process and requires interaction between the data being produced and the models. That is, as datasets are dynamically updated, the corresponding model is refined by processing and analyzing the newly collected data. In addition, new and refined models can drive what and how new data should be collected (e.g., by running new simulations or executing new experiments). Knowledge discovery and data mining is a key component in such dynamic data-driven application systems (DDDAS). It is concerned with deriving interesting, non-obvious, and useful patterns from large volumes of data. While database technology has

provided us with the basic tools for accessing and manipulating data-stores, providing interactive response times in a DDDAS, however, is challenging for several reasons.

First, the I/O requirements of data mining queries are very high, often requiring multiple accesses to large stores of data, thanks to advances in sensor and computation technologies that enable generation of very large datasets. For example, many companies already have data warehouses in the terabyte range (e.g., FedEx, UPS, Walmart). Similarly, scientific data is reaching gigantic proportions (e.g., NASA space missions, genomic data banks).

A cluster of nodes with high speed interconnects and high-capacity commodity disks, can create *active storage nodes* that enhance the ability to store, preprocess, and manipulate large-scale scientific and business data. These characteristics of commodity clusters make them cost-effective and viable to be *end-nodes* in the Data Grid [14] and to form the building blocks for next generation data centers with mass storage systems serving very large datasets. We expect that such data centers will be increasingly ubiquitous along with increasing cost-effectiveness of spinning storage and that such centers will be an essential cog of future Data-Grid and Knowledge-Grid architectures [14, 8]. Grid middleware technologies, such as Globus [21] and Storage Resource Broker [47], enable access to storage and compute end-nodes across multiple administrative domains. This decentralization of storage of data has the potential to improve access time to data. However, data management and manipulation is complicated by the distributed, and inevitably heterogeneous nature of such data analysis centers. In this kind of environment, data management and manipulation must be coordinated through a middleware framework that exists between data sources and applications.

Second, upon a data update, or changes to the mining technique's input parameter set, the set of valid patterns will change. The naive approach of simply re-executing algorithms from scratch on a database update can result in an explosion in the computational and I/O resources required. Caching and re-using previously mined information or previously accessed data cached will improve the scalability of the process. Effectively re-using information available in the cache is often technique specific; thus this process has

* This research was primarily supported by the National Science Foundation under Grant #NGS-CNS-0406386. The authors would also like to acknowledge National Science Foundation Grants #ITR-CNS-0426241 and #RI-CNS-0403342.

to be developed in coordination with the application developer. In addition, data mining becomes more complicated in the context of dynamic databases, where there is a constant influx of data.

Third, the computational requirements of such data mining queries are also very high. Scalable parallel implementations are essential. Given the current hardware advances, clusters built from off-the-shelf CPUs and high-speed interconnects will provide cost-effective *compute nodes* for computation-intensive data mining applications. These clusters are often heterogeneous in nature, resulting in an imbalance in ability among nodes. There is also the need for balancing the needs of the clients' requests with dynamic resource availability. Thus, scheduling strategies that take into account the dynamic and heterogeneous nature of the environment must be developed hand-in-hand with the characteristics of data mining techniques.

Finally, there is a need for redesigning existing data mining algorithms to take advantage of such systems support. The redesigned algorithms will need to interact with the various system components in order to specify data requirements, potential re-use patterns and determine alternative paths of execution should the need arise (due to resource constraints).

In this work, we present a software framework for deploying data analysis centers designed for KDD algorithms on dynamic datasets. This framework targets distributed storage and compute cluster environments and consists of a suite of services and distributed runtime system (see Figure 1). This paper describes the overall architecture of our framework and its services. We also present the design of a sampling service for dynamic data, implemented as part of the data services, with experimental results showing the benefits of our sampling strategies.

2. Motivating Applications

Knowledge discovery and data mining is ultimately motivated by the need to analyze data for a variety of practical applications. The proposed framework can facilitate execution for a wide range of applications. Examples include analyzing network flow data for security and accounting, analyzing e-commerce transactions for customer profiling and targeted advertising, analyzing online documents and email traffic for security and spam-control purposes, and finally analyzing scientific data produced by simulations to enable novel scientific discoveries.

The specific applications we are targeting include network monitoring [27], effective management of oil reservoirs [43], and molecular dynamics simulation data analysis [15, 24]. As our sampling service (to be discussed in sec-

tion 5) is motivated by network monitoring applications, we will detail the needs of this application below.

Network monitoring applications analyze network flow data that is generated by routers, with the goal of finding intrusions in an online fashion. The network flow logs generated by these routers are typically high dimensional and these need to be processed in real-time. Recent work in intrusion detection has seen the use of frequent pattern mining [27, 2, 3, 45] for signature detection, the use of clustering for unsupervised anomaly detection[39, 18], and the use of classification techniques[2, 3] for intrusion detection. These approaches however have largely been offline and sequential.

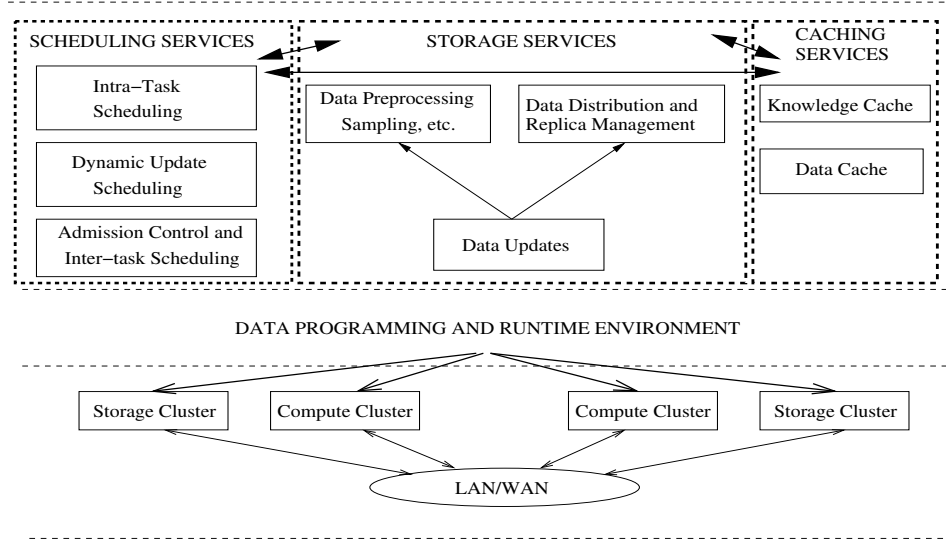
In order facilitate real-time intrusion detection, said algorithms need to be able to process network data as and when it arrives. Re-executing the algorithm over the entire data set from scratch is not an option and this motivates the development of incremental data mining algorithms. Intrusion detection algorithms are typically compute intensive, and in order to facilitate real-time processing, both parallel and incremental intrusion detection algorithms are needed so as to scaleup performance. Network traffic being bursty, the data mining workload at the analysis center can change over time. Consequently, load aware scheduling strategies for parallel incremental intrusion detection, to cope with changing network traffic, are needed.

Samples are often used to scale up the performance of I/O intensive data mining algorithms. Samples are also used by a network administrator to summarize essential parts of network traffic. Several outlier detection algorithms can use sampling to effectively summarize distributed sources of data [20]. Thus, one needs the ability to pose adhoc sampling queries. A sampling service capable of efficiently answering sampling queries is needed. Extant approaches need time proportional to data set size when generating a sample (details in section 5), which is expensive, and approaches that produce a sample in time proportional to the sample size are needed.

3. System Architecture

The overall architecture of our framework is shown in Figure 1. We target a services-oriented framework to take advantage of emerging Web and Grid services standards for interoperability with other frameworks. Moreover, a services-based framework offers an agile environment, where individual services can be customized and optimized for a particular application and the overall system can be easily extended by adding new services. In this section, we describe the core services and the underlying runtime support of the middleware framework.

DATA MINING APPLICATIONS BUILT ON TOP OF DATALET RUNTIME LAYER



3.1. Datalet Runtime Support

The hardware configurations targeted in this work are heterogeneous, distributed storage and compute systems. To support efficient execution in such an environment, the underlying runtime support for the core services is built on light-weight processes, referred to as *datalets*. A datalet is a light-weight object customized for user-defined processing. The datalet infrastructure draws from the previous work in Active Disks [49] and DataCutter [7]. Building on similar principles, the datalet environment supports use of both task and data parallelism. The interaction of a datalet both with other datalets and with the environment is done through logical pipes. Logical pipes denote uni-directional flow of data and control information. We define two types of pipes; control pipes are used to describe the flow of control information between two datalets or between the application program and a datalet, and data pipes represent data streams between two datalets or between a datalet and the storage system. Data is delivered as a sequence of named buffers through streams in a pipelined fashion. The sources and sinks for the input and output pipes of a datalet are specified by the application during instantiation of the datalet. These pipes are then fully managed by the datalet runtime environment. A datalet can write to and read from a pipe using the system functions *put()* and *get()*.

Our system provides two types of datalets. *Application-level datalets* carry out user-defined data mining computations. *System-level datalets* perform common operations, such as replication, caching, and de-clustering. The runtime environment is responsible for automatic memory management for datalets. In addition, it coordinates datalet scheduling both in relation to each other and in relation to resource availability.

3.2. Storage Services

The data storage service implements strategies for managing and accessing data in distributed disk space. It also supports basic update-propagation trigger services for dynamic datasets. Update propagation is relatively straightforward in nature. Basically, data mining applications which operate on dynamic datasets will register with this service. These applications must identify their data source(s), and provide handles to needed service routines. The paradigm is similar in nature to the InterAct framework [36].

Storage Services support data replication and de-clustering. In order to achieve high I/O rates, organization and distribution of data across the system is paramount. The objective of data distribution is to balance the storage and I/O load across the nodes so that the maximum aggregate disk bandwidth can be used, while minimizing communication overhead. The performance of a particular data distribution depends on data access patterns and the dataset being accessed. With dynamic datasets, as new data is added to a dataset, the dataset may need to be redistributed. In some cases, query workloads may have a diverse set of query types (e.g., queries accessing the entire dataset, queries focusing in on a particular set of attributes or range of attribute values), which may exhibit a variety of access patterns. In this case, portions of the dataset can be replicated and redistributed based on the types of queries. In our framework, a *planner* datalet computes an I/O and communication schedule for the data based on the output of the data replication and distribution algorithm. The goal of the planning phase is to compute a schedule which minimizes data transfer time. Finally, a *data mover* datalet is responsible for executing the schedule and transferring the data.

Many data mining algorithms rely on key data preprocessing techniques that are better suited for computation at or near the storage device. Methods of particular interest are dimensionality reduction techniques and sampling methodologies. Often real data shows considerable inter-attribute correlation. Based on this observation researchers have noted that in many real datasets the number of implicit dimensions is only a small fraction of the original dimensions [33]. Therefore, one solution is to create a representation of the data possessing a fewer number of dimensions [23]. Using dimensionality reduction techniques to represent dynamic datasets is complicated due to the possibility that conceptual directions (as given by the principal components) may drift [17].

Sampling has often been suggested as an effective tool to scale the performance of database operations [28]. Recently, this strategy has been applied to data mining algorithms [35]. By reducing the size of the dataset (in this case, by eliminating rows of the dataset) operated on, one is able to reduce I/O and computational costs at some cost to accuracy. Mining on a sample rather than the complete data is an effective strategy for adapting to limited system resources. We present the design of our sampling service in Section 5.

3.3. Caching Services

Caching services provide support to maintain duplicate data and result summaries for reuse. Data mining operations benefit from two kinds of data and computation reuse strategies. *Data caching*, which caches subsets and summaries of datasets used across multiple data mining operations, is a general-purpose caching strategy. The data cache consists of a memory cache and a disk-based persistent cache. The persistent cache augments the memory cache in two ways. First, it can be used as overflow cache for items that are evicted from the memory cache. When a new data subset is cached in memory and memory allotted for the data cache is full, the cached data subset evicted from the memory cache is staged to the disk-based persistent cache. Second, it enables data reuse across multiple invocations of the data cache by maintaining cached results on disk.

The caching service also implements a *data-mining technique centric knowledge cache*, containing recently constructed and/or used knowledge *objects*, that can be leveraged along with our data cache. Knowledge caching caches meta-level summaries or results that are created by prior queries and that are specific to a given mining technique (e.g. increment sequence lattices for interactive sequence mining [38]).

Both strategies are orthogonal in nature and can be used in conjunction for a given query. An incoming data mining query can be evaluated against both the data cache and the knowledge cache to determine if any of the cached re-

sults can be reused and to find cached data subsets that are needed to answer the query.

In our framework, both the data cache and the knowledge cache are implemented as a set of system level datalets which interact with application level datalets. A cache datalet is responsible for managing both local disk space and memory space allocated for cache on a machine. A group of cache datalets form a distributed cache that uses space dispersed across multiple nodes. These datalets implement adaptive strategies for placement and eviction for the system. In continuous data mining queries, the summary is incrementally maintained across data updates. Thus, cached objects will also need to be updated periodically, to maintain coherency. Our principle strategy to support such *dynamic cache objects* is to replace the old version with the new version at periodic time intervals. In the InterAct project we handled this in an efficient manner by adopting twinning and diffing [36].

3.4. Scheduling Services

The scheduling services address admission control of data mining jobs, and intra-task scheduling for both static and dynamic data mining tasks. In our framework, the scheduling service also provides support for automated execution of system level datalets for data distribution, summary computation, data replication, and data caching.

When a new application task request is to be scheduled for execution, the task can be scheduled onto all available machines in the system. However, the main disadvantage of this approach is that an expensive task may consume most of the resources, thus adversely affecting the performance of other requests submitted to the system. If the processing needs of the data mining job can be met by available resources in the system, the job will be admitted and assigned to a specific set of workstations.

A data mining application can consist of a sequence of data and compute intensive operations, implemented as datalets, on data. Each application datalet performs a piece of the application data processing. These datalets can be organized into a collective to implement the entire processing structure for a particular data mining application. In this collective, data is processed as it is retrieved from data sources and moved to the client. This set of datalets along with the resources they require (either cached data or raw data) as well as any interdependencies among them are given to the intra-task scheduling service for scheduling. The datalet support enables combined use of task- and data-parallelism in implementing a data mining application. In order to decrease the execution time of a data mining application, the intra-task scheduling service creates and remove copies of datalets, effectively modifying the datalet network of the application and place the copies on storage and computa-

tion platforms to minimize communication and computation overheads.

The scheduling service also implements algorithms for scheduling multiple operator graphs from jobs concurrently submitted to the system. In [46], we looked at strategies for adaptive task- and data-parallelism for multiple data visualization applications implemented as a pipelined sequence of operators. Our results showed that good performance can be achieved by mutating the operator graph and carefully placing copies of operators on machines.

When new data is received, the scheduling service inspects which datasets, replicas, and cached objects are affected by the data update. If there are jobs and sub-tasks that are waiting to be scheduled which do not require these objects, those jobs and sub-tasks will be scheduled onto the available resources. The system level datalets are then scheduled onto the remaining resources. Some system level operations such as data distribution or replication can be terminated and restated as needed.

4. Related Work

Large data grid projects should be layered on robust and efficient middleware systems. As a result, a large body of research has been targeted at developing middleware frameworks, protocols, and programming and runtime environments [21, 19, 47, 52, 6, 11, 32, 16, 51, 44, 9, 48, 29, 5]. These efforts mainly focus on services for resource allocation, unified file access, replica management, distributed execution, security and authentication. Recently, the emerging class of data-intensive applications got the attention of several research groups resulting in publications on high-speed transport [1], replica management [13, 42], meta data and catalog management [4]. We plan to leverage these middleware tools in our work.

Several researchers have looked at the problem of systems support for distributed data mining algorithms. JAM[40], BODHI[26] and Info-sleuth[30] are high level agent-based distributed systems (in user-space) that support such algorithms. Similar to this work, the Papyrus[22], Kensington[12], FreeRIDE[25], and Intelliminer[37] (developed by the PI) systems look at the problem in terms of clients, and compute and storage servers and implemented basic system level services for data transfer and scheduling of parallel tasks. None of these systems are designed to handle dynamic data updates, or support high performance storage, caching and advanced scheduling services as outlined in this work. The Data-Space system, currently being developed, shares some aims with the storage services of our middleware although it is directed at a more wide-area context. The current project leverages off the InterAct system (developed by the PI) [36] which was designed to support shared state for client-server data mining applications in a wide-area distributed environ-

ment. Recently, several groups have been looking at mining over a grid environment, but most of this work is quite preliminary in nature[8, 10, 50, 31].

5. Data Processing - Sampling Service

Mining data is typically an iterative process, requiring multiple passes over the data set, which may be prohibitively expensive. These problems are exacerbated when data is streaming at a high rate, and must be processed in near real time. As a result, researchers in the data mining field have increasingly turned to sampling as a means of compromising quality for improved response times. In some cases, such as anomaly detection, ad-hoc queries with sampling can be used to summarize data for applications [20].

Despite the recognized importance of sampling in data mining and data analysis, very little work has been done in making sample generation efficient. As noted by Provost and Kolluri [41], "most discussions on sampling assume that producing random samples efficiently from large data sets is not difficult. This is simply not true." In fact, most implementations require $O(N)$ time where N represents the size of the data set and not the sample size (S). To illustrate this problem, consider sampling from a transactional database. Let us assume that the average size of the transaction is roughly 40 bytes, and a disk block is 4 kilobytes (producing about 100 records per block). Sampling at 1% or higher results in accessing every block. Therefore, each sampling pass is equivalent to one scan of the entire database. An architecture to efficiently support such sampling queries for next generation data analysis applications is the focus of this work. Specifically, we address the following challenges:

- Emerging application domains such as network intrusion detection need to process out of core data sets that are dynamic and large in volume. Samples of network transactions are often used to build a model for normal behavior to be able to detect intrusions in real time.
- The iterative nature of data analysis algorithms requires the ability to generate variable size samples. Certain applications demand sample size be progressively increased, until some quality criterion has been satisfied. Furthermore, the sample may not be a random sample over the entire stream, but rather a sample of a certain historic time range. We need the ability to generate samples in which both the size of the sample and the time range of the sample can be specified.
- With advances in networking technology, high bandwidth interconnects such as those provided by Infiniband make network I/O faster than disk I/O. This property can be used to reduce the sample generation time

by parallelizing the sample generation process across parallel disks.

Our sampling strategies address the above challenges. Our primary contributions are:

- We have designed a flexible query system. For example, our query service supports queries of the form

```
SELECT < attributes > FROM Dataset
WHERE time_start < time < time_end
SAMPLE x%
```

Here, x is the sample size as a percentage of the data selected by the predicate $time_start < time < time_end$.

- We have developed an intelligent online data placement strategy to enable efficient querying and sample generation on dynamic datasets. Our approach relies on the notion of stream windows and bins. A stream window corresponds to a user defined number of transactions (n) on the incoming data stream. Each window of n transactions is further partitioned into a user defined number of bins (k). The algorithm uses a randomization function such that each bin can be viewed as mutually exclusive samples of the stream window. Essentially, each incoming transaction is assigned to one of the k bins with probability proportional to the size of the bin. Placement within a bin is ordered according to arrival by default, preserving temporal ordering. However, a user-specified hash function may be specified to order transactions within a bin. Such a hash-based ordering scheme can be leveraged to support stratified or periodic sampling, which has been used effectively when modeling network traffic [34]. This process is repeated for every n transactions. Each bin can be of a fixed size. Typically, however, it is of variable size, following a geometric progression ($n/2, n/4, n/8$ and so on). As motivated in the introduction, data analysis sampling queries often involve sample requests of variable sizes. Geometrically progressing bin sizes allow efficient and quick processing of such multi-resolution sampling query requests. Each bin is also placed contiguously on disk. This ensures that when extracting a sample, if a bin of size equal to the sample size is chosen, the total cost of generating the sample will be aptly proportional. If the desired sample cannot be retrieved using a single bin, the overhead in disk I/O is bounded by the size of the smallest bin in the stream window. If a sample request spans multiple stream windows, then samples from each one can be appropriately combined to derive the requested sample.
- When multiple disks are available, we adopt the following round robin placement strategy. Assume we

have m parallel disks. Let $\text{bin}(r, i)$ denote bin i within stream window r . Bin distribution is an onto mapping f from the bin set $\{0, 1, \dots, k - 1\}$ to the parallel disk set $\{0, 1, \dots, m - 1\}$ such that bin id (r, i) will be assigned to node $f_r(i)$ as follows: Given a mapping f_{r-1} for window $(r - 1)$, f_r will be determined by formula

$$f_r(x) = [f_{r-1}(x) + 1] \% m \quad (1)$$

We call this "round-robin bin distribution". Since we are defining f in an iterative way, f_0 is needed for the first bin placement. One can simply use:

$$f_0(i) = i \% m. \quad (2)$$

Basically, our formulation says if a bin (r, i) ($0 \leq i < k$) is assigned to disk y , then bin $(r + 1, i)$ will be assigned to next disk in order, namely $(y + 1) \% m$. This bin distribution strategy has the following two properties:

1. $f_i = f_{i+m}$; after every m windows, $m \times n$ elements will be equally distributed onto m disks.
2. Given m disks, any bin with id (r, i) will be on disk k if and only if $(r + i - k) \% m = 0$.

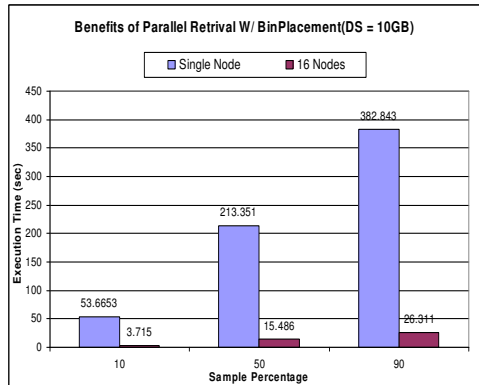
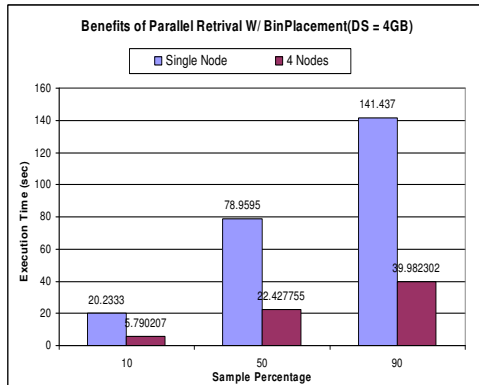
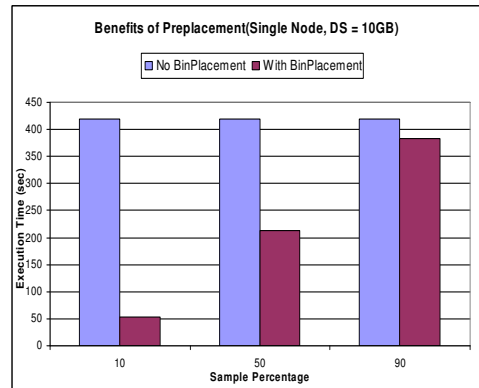
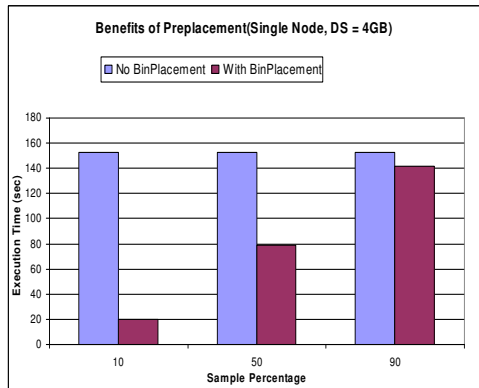
Thus, given a bin identifier we can determine in constant time where to place it.

We have evaluated the benefits of our placement strategy for efficient parallel sample retrieval in a cluster setting. The benefits of our online placement strategy on a single node are evident in Figure 2. On two large datasets, the time required to retrieve a sample is proportional to the sample size rather than to the size of the dataset. Variable sized random samples are retrieved by touching disk-blocks proportional to the size of the sample rather than the size of the dataset.

On a cluster of storage nodes, round robin sample placement and retrieval permits excellent scaling with the number of nodes. Our empirical results (Figure 3) show that on the cluster with Infiniband the scaleup is as expected, with a speedup of 3.6X on four nodes and 14X on sixteen nodes.

6. Conclusion

In this paper, we present our vision of a system to support large scale dynamic data driven applications. It is our belief that current applications require support for operations on large, dynamic data sets. While the framework is general-purpose in this regard, we have outlined strategies to specifically support knowledge discovery and data mining applications. In addition, we have detailed a portion of the current implementation of our sampling services. We have shown that it significantly improves I/O performance of data mining algorithms. Our near-term future work will



target caching services, as described in section 3.3. In addition, we are in the process of developing parallel and incremental variants of various data mining algorithms used in intrusion detection.

References

- [1] B. Allcock, J. Bester, J. Bresnahan, A. L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnal, and S. Tuecke. Data management and transfer in high performance computational grid environments. *Parallel Computing Journal*, 28(5):749–771, May 2002.
- [2] D. Barbara and S. Jajodia, editors. *Applications of Data Mining in Computer Security*. Kluwer, 2002.
- [3] D. Barbara, N. Wu, and S. Jajodia. Detecting novel network intrusions using bayes estimators. In *Proceedings of SIAM Data Mining*, 2001.
- [4] C. Baru, R. Moore, A. Rajasekar, and M. Wan. The sdsc storage resource broker. Toronto, Canada, 1998.
- [5] F. Berman and R. Wolski. The AppLeS project: A status report. In *Proceedings of the NEC Research Symposium*, Berlin, Germany, May 1997.
- [6] M. D. Beynon, T. Kurc, U. Catalyurek, C. Chang, A. Sussman, and J. Saltz. Distributed processing of very large datasets with DataCutter. *Parallel Computing*, Oct. 2001.
- [7] M. D. Beynon, T. Kurc, A. Sussman, and J. Saltz. Optimizing execution of component-based applications using group instances. In *IEEE CCGrid*, May 2001.
- [8] M. Cannataro and D. Talia. Knowledge grid an architecture for distributed knowledge discovery. In *CACM*, 2003.
- [9] H. Casanova and J. Dongarra. Applying Netsolve’s network-enabled server. *IEEE Computational Science & Engineering*, 5(3):57–67, July-September 1998.
- [10] R. J. Cavanaugh. Sphinx: A Scheduling Middleware for Data Intensive Applications on a Grid. In *Workshop on Data Mining and Exploration Middleware for Distributed and Grid Computing*, 2003.
- [11] Common Component Architecture Forum. www.ccaforum.org.
- [12] J. Chatratchat, J. Darlington, Y. Guo, S. Hedvall, M. Koller, and J. Syed. An architecture for distributed enterprise data mining. In *HPCN europe*, pp.573-582, 1999.
- [13] 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. Giggie: a framework for constructing scalable replica location services. In *Proceedings of ACM/IEEE SC02*, 2002.
- [14] 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 Datasets, 2001.
- [15] M. Coatney and S. Parthasarathy. Motifminer: A general toolkit for efficiently identifying common substructures in molecules. In *IEEE BIBE*, 2003.
- [16] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. Grid information services for distributed resource sharing, 2001.
- [17] P. Domingos, L. Spencer, and G. Hulten. Mining time changing data streams. In *Proceedings of ACM SIGKDD*, 2001.

- [18] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. S. o. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. In D. Barbara and S. Jajodia, editors, *Applications of Data Mining in Computer Security*. Kluwer, 2002.
- [19] J. Frey, T. Tannenbaum, I. Foster, M. Livny, and S. Tuecke. Condor-G: A computation management agent for multi-institutional grids. In *Proceedings of IEEE HPDC*, 2001.
- [20] A. Ghoting, M. Otey, and S. Parthasarathy. LOADED: Link-based outlier and anomaly detection in evolving datasets. In *Proceedings of IEEE ICDM*, 2004.
- [21] The Globus Project. www.globus.org.
- [22] R. Grossman, S. Bailey, A. Ramu, B. Malhi, and A. Turinsky. The preliminary design of papyrus: A system for high performance, distributed data mining over clusters, meta-clusters and super-clusters. In *Advances in Knowledge Discovery and Data Mining*, 2000.
- [23] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.
- [24] M. Jiang, T.-S. Choy, S. Mehta, M. Coatney, S. Barr, K. Hazzard, D. Richie, S. Parthasarathy, R. Machiraju, D. S. Thompson, J. Wilkins, and B. Gatlin. Feature Mining Algorithms for Scientific Data. In *Proceedings of SIAM Data Mining*, May 2003.
- [25] R. Jin and G. Agrawal. A middleware for developing parallel data mining implementations. In *Proceedings of SIAM Data Mining*, Apr. 2001.
- [26] H. Kargupta, B.-H. Park, D. Hershberger, and E. Johnson. Collective Data Mining: A New Perspective Towards Distributed Data Mining. In H. Kargupta and P. Chan, editors, *Advances in Distributed Data Mining*. AAAI Press, 2000.
- [27] W. Lee and S. J. Stolfo. A framework for constructing features and models for intrusion detection systems. *Information and System Security*, 3(4):227–261, 2000.
- [28] Y. Ling and W. Sun. An evaluation of sampling-based size estimation methods for selections in database systems. In *IEEE ICDE*, pages 532–539, 1995.
- [29] Logistical Computing and Internetworking Lab. loci.cs.utk.edu/.
- [30] P. Martin, V. Callaghan, and A. Clark. High performance distributed objects using caching proxies for large scale applications. In *International Symposium on Distributed Objects and Applications*, 1999.
- [31] C. Mastroianni, D. Talia, and P. Trunfio. Managing Heterogeneous Resources in Data Mining Applications on Grids Using XML-Based Metadata. In *IPDPS*, 2003.
- [32] R. Oldfield and D. Kotz. Armada: A parallel file system for computational. In *Proceedings of IEEE CCGrid*, May 2001.
- [33] B. U. Pagel, F. Korn, and C. Faloutsos. Deflating the dimensionality curse using multiple fractal dimensions. In *Proceedings of IEEE ICDE*, 2000.
- [34] J. Pan, C. Faloutsos, and S. Seshan. Fastcars: Fast correlation aware sampling for network data mining. In *IEEE GlobeCOM*, 2002.
- [35] S. Parthasarathy. Efficient progressive sampling for association rules. In *Proceedings IEEE ICDM*, 2002.
- [36] S. Parthasarathy and S. Dwarkadas. Shared state for distributed interactive data mining applications. *Kluwer International Journal on Distributed and Parallel Databases*, 2002.
- [37] S. Parthasarathy and R. Subramonian. Facilitating data mining on a network of workstations. In *Advances in Distributed and Parallel Knowledge Discovery*, 2000.
- [38] S. Parthasarathy, M. Zaki, M. Ogihara, and S. Dwarkadas. Incremental and interactive sequence mining. In *Proceedings of ACM CIKM*, Mar 1999.
- [39] L. Portnoy, E. Eskin, and S. Stolfo. Intrusion detection with unlabeled data using clustering. In *Proceedings of ACM CSS Workshop on Data Mining Applied to Security*, 2001.
- [40] A. L. Prodromidis, P. K. Chan, and S. Stolfo. Meta-learning in distributed data mining systems: Issues and approaches. In *Advances in Knowledge Discovery and Data Mining*, 2000.
- [41] F. Provost and V. Kolluri. A survey of methods for scaling up inductive algorithms. *Knowledge Discovery and Data Mining*, 3, 1999.
- [42] K. Ranganathan and I. Foster. Identifying dynamic replication strategies for high performance data grids. In *Proceedings of International Workshop on Grid Computing*, 2002.
- [43] J. Saltz and et.al. Driving scientific applications by data in distributed environments. In *Dynamic Data Driven Application Systems Workshop, held jointly with ICCS 2003*, 2003.
- [44] M. Sato, H. Nakada, S. Sekiguchi, S. Matsuoka, U. Nagashima, and H. Takagi. Ninf: a network based information library for a global world-wide computing infrastructure. In *Proceedings of HPCN'97 (LNCS-1225)*, 1997.
- [45] K. Sequira and M. Zaki. ADMIT: Anomaly-based Data Mining for Intrusions. In *SIGKDD Conference*, 2002.
- [46] M. Spencer, R. Ferreira, M. Beynon, T. Kurc, U. Catalyurek, A. Sussman, and J. Saltz. Executing multiple pipelined data analysis operations in the Grid. In *ACM/IEEE SC02*.
- [47] SRB: The Storage Resource Broker. www.npaci.edu/DICE/SRB/index.html.
- [48] D. Thain, J. Basney, S. Son, and M. Livny. Kangaroo approach to data movement on the grid. In *Proceedings IEEE HPDC*, 2001.
- [49] M. Uysal, A. Acharya, and J. Saltz. Evaluation of active disks for decision support databases. In *Proceedings of HPCA*. 2000.
- [50] V. Čurčín, M. Ghanem, Y. Guo, M. Köhler, A. Rowe, J. Syed, and P. Wendel. Discovery Net: Towards a Grid of Knowledge Discovery. In *Proceedings of the ACM SIGKDD*, 2002.
- [51] S. Vazhkudai, S. Tuecke, and I. Foster. Replica selection in the globus data grid, 2001.
- [52] R. Wolski, N. Spring, and J. Hayes. The network weather service: A distributed resource performance forecasting service for metacomputing. *Journal of Future Generation Computing Systems*, 15(5-6):757–768, 1999.