The National Scalable Cluster Project: Three Lessons about High Performance Data Mining and Data Intensive Computing

Robert Grossman[1] and Robert Hollebeek[2]

[1] University of Illinois at Chicago and Magnify Inc.
[2] University of Pennsylvania and Hubs Inc.

1. Introduction

Today a 10 GB data set can easily be produced with a simple mistake, 100 GB - 1 TB data sets are common, and data sets larger than a Terabyte are becoming common. In this paper, we describe three basic lessons we have learned about data mining and data intensive computing large and massive data sets. First, we give a few definitions. A Terabyte is 1000 Gigabytes and a Petabyte is a 1000 Terabytes. A data set is large if it is larger than most of your friends' data sets. A data set is massive if it is larger than most other data sets you have seen. We assume the data is divided into files, that files are divided into records, and that records contain attributes. By archiving data, we mean storing data in such a way as to provide file-level access to the data. By managing data, we mean storing data in such a way as to provide record-level and attribute-level access to the data. By mining data, we mean looking for patterns, changes, associations, anomalies and other statistically significantly structures in data.

We employ the following three very rough rules of thumb:

> Rule 1. You can *archive* 10x-1000x more data with a data archival system than you can *manage* with a data management system, and you can manage 10x-1000x more data with a data management system than you can *mine* with a data mining system.
>
> Rule 2. Simple ideas scale the best.
>
> Rule 3. People usually claim that they can mine 10x-1000x more data than they can by misunderstanding Rule 1.

Rule 1 is a consequence of how systems are designed and how data is moved. A system can index and efficiently access a certain number of elements. For archival systems, these elements are files; for databases systems, these elements are records. Files are typically much larger than records. Most data mining algorithms require scanning or moving the data multiple times, which explains the third part of the Rule 1, although some algorithms require only a few scans of the data. Rule 2 is a reflection of how software and systems are typically built. Rule 3 is a consequence of how people are designed.

The National Scalable Cluster Project (NSCP) is a NSF-funded collaboration focused on data mining and data intensive computing of large and distributed data sets. The NSCP has exploited the fact that beginning approximately in the mid-1990s workstations were becoming commodities. By forming local clusters of workstations with high performance switches and hubs and by using the appropriate data and compute management software, super-computer scale data intensive computing became practical using clusters. The NSCP-1 Meta-Cluster was completed in 1996 and linked geographically distributed clusters using the commodity Internet. The NSCP-2 Meta-Cluster was completed in 1998 and used OC-3 networks to link the clusters.

In this paper, we describe three lessons about data mining and data intensive computing we have learned while working with the NSCP-1 and NSCP-2 Meta-Cluster. Section 2 contains a brief description of the NSCP Meta-Cluster. Section 3 describes some related work. Sections 4-6 describe the lessons and Section 7 is the conclusion.

2. The NSCP Meta-Cluster

The National Scalable Cluster Project (NSCP) collaboration of research groups has pioneered the application of cluster computing and high performance wide area networks to a variety of problems in data mining and data intensive computing, including working with several terabyte size collections. The core research groups from the University of Illinois at Chicago and the University of Pennsylvania work

with collaborators at over ten institutions. The project was founded in 1994 by Grossman and Hollebeek.

The NSCP collaborators have assembled local clusters of workstations and connected these local clusters into wide area clusters of clusters or *Meta-Clusters*. NSCP also developed several software packages for data intensive computing using the Meta-Cluster. The NSCP-1 Meta-Cluster was completed in 1996 and linked geographically distributed clusters using the commodity internet. The NSCP-2 Meta-Cluster was completed in 1998 and used OC-3 networks to link the clusters. The NSCP-1 and NSCP-2 Meta-Clusters have been used by a variety of scientists and engineers working on applications in high energy physics, computational chemistry, nonlinear simulation, bioinformatics, medical imaging, network traffic analysis, digital libraries of video data, and economic data.

An NSCP-3 Meta-Cluster is currently being designed and tentatively scheduled for deployment in 2001. The NSCP-3 Meta-Cluster is being designed to exploit wave division multiplexing (WDM) technology. WDM is now being used to greatly increase the available bandwidth on links connecting geographically distributed nodes by packing many wavelengths carrying separated data streams onto a single fiber.

Currently, the NSCP consists of approximately 100 nodes and 3 terabytes of disk geographically distributed among the participating sites, and connected by laboratory, campus, and national ATM networks. NSCP developed software and third party software are provided so that applications can transparently access as many nodes and as much disk as required. NSCP supports one large digital library (500 Gigabytes), two moderate size digital libraries (100 Gigabytes each), and several smaller ones. More details can be found at http:/nscp.upenn.edu and http://www.nscp.uic.edu.

An Introduction to HP-D2M2: High Performance Distributed Data Mining and Management

The term "data mining" is often used to refer narrowly to the data-intensive and compute-intensive *process* of winnowing nuggets of useful information from a data source. We think it is helpful to include in that catch phrase the entire required sequence of processes including system design, system operation and optimization, as well as the algorithm design and the software design required for the grand goal of the process — namely knowledge discovery. Our own interest in data mining extends primarily to its application to very large and/or distributed collections of information. Since the goal of data mining is to extract important and — one hopes —previously unknown information from such collections, doing this quickly and efficiently, or doing it on very large distributed and often disorganized collections of data requires high performance techniques. We focus here on three lessons that we have learning in the past several years through our work in the National Scalable Cluster Project, and the Terabyte Challenge, applying new techniques to large distributed data samples.

The first lesson is that we must parallelize the hardware and software I/O in data intensive computing problems and that in doing so, successive layers of hardware and software need to be well matched to potential choke-points. The second level is that we need to exploit data parallelism, and the third and final lesson is that we must pay particular attention to data layout in both hardware and software.

The phenomenon of rapidly increasing volumes of data is familiar to all by this time. Anyone with a home PC, an office machine, a portable, and perhaps a palm device knows something about how rapidly information can become diffuse and disorganized. Just as we do not always have good methods of organizing our personal piles of paper, or the scraps on our desks, we do not always have effective means of organizing our large distributed digital collections either. As the total volume of information increases, and as it becomes more geographically dispersed, the process of data mining can become very difficult.

The first — and perhaps most important — step in the data mining and knowledge-discovery process is to select and locate *relevant and interesting* data. We then need to accumulate, aggregate, organize and catalog it. The goal of the process of data mining is to extract **new** knowledge. Given the large number of potential avenues of investigation, the next step is to automate or semi-automate the extraction of features, clusters, aggregates or other interesting correlations among parts of the data. Fortunately, high performance computers can search through incredible numbers of combinations and possible scenarios even for large collections of data.

Data mining exercises are highly iterative and require human intervention between iterations. Often, knowledge obtained in a first iteration must be fed back into the system, and the next iteration steered in a new direction. Anyone who hopes to simply pour wheat chaff (i.e., raw data) in the top funnel of a giant vat, and sip Cognac dripping from the bottom in a hands-off, completely automated fashion, is setting their sights too high. The best data mining process will require thoughtful collaboration between content experts and experts in the data mining process.

Almost any collection of information has the potential of providing new knowledge, and in some sense, the more information that is collected, the more likely new knowledge is. There is a temptation to shy away from very large collections due to the technical difficulties of managing and processing them. So the major goal is to recognize the value of growing large collections of information, and then to design new ways in which the relationships contained in that information might be extracted and used.

There are many potential choices for what data one might collect and mine, and it is very important to give some careful thought to selecting a set of information which, at least initially, is thought to contain some useful, valuable and actionable information. It makes little sense to start with dull and uninteresting data since there is no magic to the data mining process even if there is a lot of black art! For beginners, there is one exception to this rule. It may be useful to pick a set of information which contains knowledge you already know, and use the data mining software in an exercise to verify this knowledge. If you have purchasing information for example, take a mix of good and bad customers (defined either in terms of purchases or support costs) and see if the data mining process can segment the information into two such classes. Then, see if it can identify some of the other distinguishing demographics of those two classes. If you are successful, you will certainly bolster your confidence in your ability to properly use the tools and interpret the results.

Once a good data sample has been selected, it is also a good idea to simplify the data by eliminating some variables, and grouping others into classes. It is always possible to carry such variables along temporarily and bring them back into the process later, but when starting out, simplicity is key. This is due, at least in part, to the rapidly rising difficulty and diminishing speed of the data mining process with the dimensionality (in terms of number of variables) and complexity of the data set. Thus, it is better to start simply, and iterate.

3. Related Work

In this section, we describe related work in cluster computing, testbeds, distributed data mining systems, and data management software. One of our model applications is mining high energy physics data. We also describe related work in this area. This section is based in part on [Grossman 1999a and 1999b].[1]

Cluster Computing. There are also a number of active research projects in cluster computing, including the Berkeley Now Project [Anderson 1995][2], the Beowolf Project [Becker 1995][3] at CalTech, and the Seamless Parallel and Distributed Computing Project [Ishikawa 1997][4] of the Real World Computing Partnership. By and large, these projects have focused on providing a general computing infrastructure for

local clusters, while the NSCP has focused on data intensive computing using wide area meta-clusters built by connecting several geographically distributed local clusters.

Testbeds. For the past five years, the NSCP-1 and the NSCP-2 Meta-Cluster have provided the infrastructure for a national data intensive and data mining testbed called the Terabyte Challenge (TC). Approximately 25 different applications have been run on the Terabyte Challenge Testbed. The Terabyte Challenge has won multiple awards and has been used as an interoperability testbed for the development of the Predictive Model Markup Language (PMML). PMML is an XML-based language for data intensive computing, which is becoming an industry standard (http://www.dmg.org).

There are a number of testbeds [5] for high performance computing, high performance networking, and distributed computing. Perhaps the most similar testbed is the Globus Ubiquitous Supercomputing Testbed Organization (GUSTO) and the Centurion Legion Project Testbed. The Globus Project, led by I. Foster and C. Kesselman, is building middleware for grid-based computation. The Legion Project, led by A. Grimsaw of the University of Virginia, is developing a wide-area system for distributed object computing. In contrast, the Terabyte Challenge Testbed has focused on two objectives a) providing an application testbed for data intensive computing using local and wide area clusters, especially those requiring high performance wide area networks, and b) a testbed for languages and protocols for data intensive computing and data mining.

Distributed Data Mining Systems. Papyrus is a layered infrastructure for high performance and wide area data mining and predictive modeling developed by the Laboratory for Advanced Computing at the University of Illinois at Chicago. Papyrus is specifically designed to support clusters of workstations, distributed clusters of workstations (meta-clusters) and distributed clusters of workstations connected with high performance networks (super-clusters).

Several other systems have been developed for distributed data mining. Perhaps the most mature are: the JAM system developed by Stolfo et. al. [Stolfo 1997][6], the Kensington system developed by Guo et. al. [Guo 1997][7], and BODHI developed by Kargupta et. al. [Kargupta:1997][8]. These systems differ in data, task and model strategies:

> *Data strategy*. Distributed data mining can choose to move data, to move intermediate results, to move predictive models, or to move the final results of a data mining algorithm. Distributed data mining systems which employ *local learning* build models at each site and move the models to a central location. Systems which employ *centralized learning* move the data to a central location for model building. Systems can also employ *hybrid learning*, that is, strategies which combine local and centralized learning. JAM, Kensington and BODHI all employ local learning.
>
> *Task strategy*. Distributed data mining systems can choose to coordinate a data mining algorithm over several sites or to apply data mining algorithms independently at each site. With *independent learning*, data mining algorithms are applied to each site independently. With *coordinated learning*, one (or more) sites coordinate the tasks within a data mining algorithm across several sites.
>
> *Model Strategy*. Several different methods have been employed for combining predictive models built at different sites. The simplest most common method is to use *voting* and combine the outputs of the various models with a majority vote [Dietterich 1997][9]. *Meta-learning* combines several models by building a separate meta-model whose inputs are the outputs of the various models and whose output is the desired outcome [Stolfo 1997]. *Knowledge probing* considers learning from a black box viewpoint and creates an overall model by examining the input and the outputs to the various models, as well as the desired output [Guo:1997]. Multiple models, or what are often called ensembles or committees of models, have been used for quite a while in (centralized) data mining. A variety of methods have been studied for combining models in an

ensemble, including Bayesian model averaging and model selection [Raftery 1996][10], stacking [Wolpert 1992][11], partition learning [Grossman 1996b][12], and other statistical methods, such as mixture of experts [Xu 1993][13]. JAM employs meta-learning, while Kensington employs knowledge probing.

Papyrus is designed to support different strategies for data-parallel, task-parallel and model-parallel computation. For example, in contrast to JAM and Kensington, Papyrus can not only move models from node to node, but can also move data from node to node when that strategy is desired. In contrast to BODHI, Papyrus is built over a data warehousing layer which can move data over both commodity and high performance networks. Also, Papyrus is a specialized system which is designed for clusters, meta-clusters, and super-clusters, while JAM, Kennsington and BODHI are designed for mining data distributed over the internet. All four systems make use of Java. JAM employs Java applets to move machine learning algorithms to distributed data. Kensington uses Java JDBC to mine distributed data. Papyrus uses Java aglets.

Moore [Moore 1998][14] stresses the importance of developing an appropriate storage and archival infrastructure for high performance data mining and discusses some of the efforts of the SDSC in this area. The distributed data mining system developed by Subramonian and Parthasarathy [Subramonian 1998][15] is designed to work with clusters of SMP workstations and like Papyrus is designed to exploit clusters of workstations. Both this system and Papyrus are designed around data clusters and compute clusters. Papyrus also explicitly supports clusters of clusters and clusters connected with different types of networks. [16]

Data Management Software. Many of the NSCP-1 and NSCP-2 Meta-Cluster applications used PTool, a light weight, high performance persistent object managed designed for local and wide area clusters and meta-clusters. PTool has successfully managed distributed terabyte sized data collections.

Data Mining Algorithms. A variety of different strategies are possible in distributed and high performance data mining and predictive modeling. Data may be moved from node to node, predictive models produced by data mining algorithms may be moved from node to node, or the results of applying data mining algorithms to specific data sets may be moved from node to node. The optimal strategy depends upon the amount of data, its distribution, and the performance characteristics of the network connecting the nodes. [17]

Classification and regression trees are standard tools in data mining. Data parallel approaches in high performance computing are also standard. A standard reference for ensembles of trees is [Breiman].

Mining high energy physics data. We have worked intensively using the NSCP-1 and NSCP-2 Meta-Clusters to develop next generation systems for mining high energy physics data. Our viewpoint has evolved over time: we began with the idea of using databases to manage HEP data (1991-1992); we then explored using specialized persistent object managers (1993-1996); we then broadened our perspective to using wide area distributed object management systems (1996-1998); more recently, we have developed a web-inspired infrastructure (1998-present). In each case, we did extensive experiments with NSCP developed software.

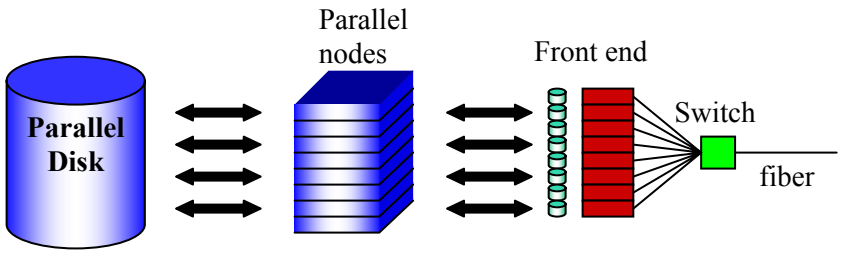
There is an extensive literature on computing with high energy physics data. Approximately every 18 months, the Conference in Computing in High Energy Physics (CHEP) meets to explore the state of the art. These proceedings contain a large number of related papers. Several projects deserve special mention. The goal of the RD-45 Project at CERN is to develop a scalable persistent object manager for petabytes of high energy physics data. The goal of the Particle Physics Data Grid led by CalTech is to develop an

infrastructure for the widely distributed analysis of high energy physics data. The goal Nile Project at Cornell is to provide a data analysis infrastructure for the CLEO detector at Cornell based upon distributed computing.

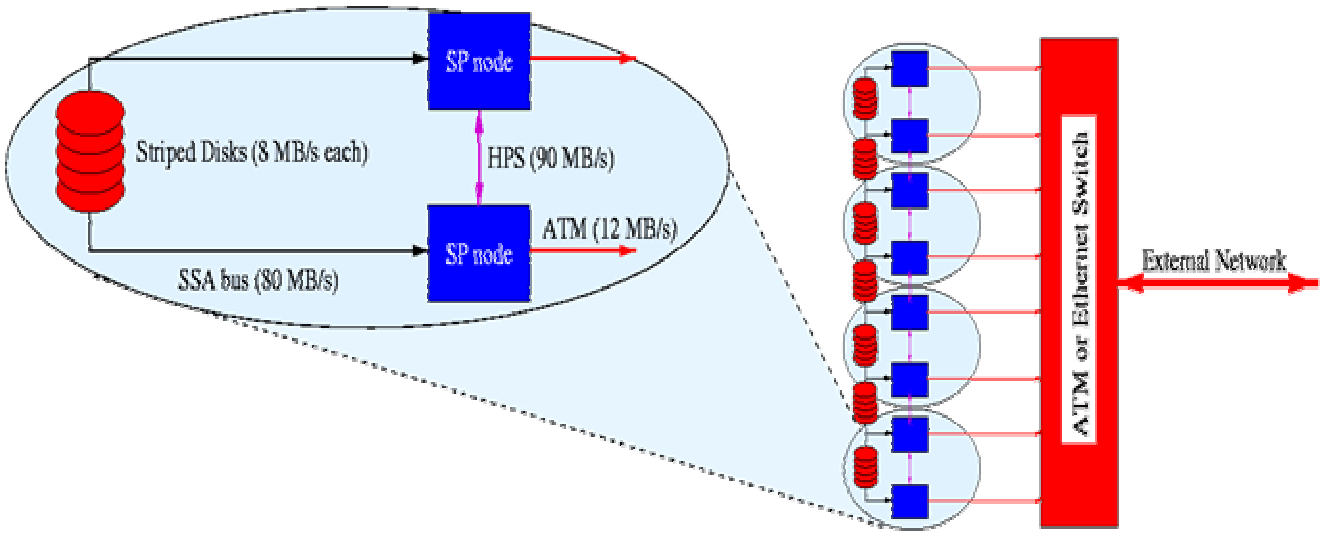
4. Lesson 1: Exploit End to End Parallelization

In this section, we wish to emphasize the first of three lessons on high performance large-scale data intensive computing that have been learning in the National Scalable Cluster Project. The first lesson is a reminder that with massive data there is no choice but to balance parallel networks with parallel input/output systems to parallel CPUs (parallel networks to parallel I/O to parallel CPUs). Think of this as optimizing the end to end parallelization from parallel I/O to parallel CPU to parallel network to parallel client.

Cluster architectures are in many ways uniquely suited to data intensive tasks. Using load balancing on parallel nodes, data striping across parallel disks and parallel execution on many nodes driven by parallel aware application programming interfaces (MPI for example), an application can be developed to take advantage of the very large *internal* bandwidth of a many-node cluster. Distributed applications, and in particular wide area distributed, data intensive applications however must deal with the bandwidths available over wide area networks (WAN) which are typically much lower than processor I/O bus and even local interconnect speeds. There are technologies emerging, such as Wave Division Multiplexing (WDM), which support parallel wide area networks so that geographically distributed clusters can be connected together in a balanced fashion. See the diagram below.



To maintain high volume data flow, systems must eliminate bottlenecks between the disk/cpu/network components. The goal is to match the throughput of each stage as the data passes from disk to IO bus to processor to the network and back again. When the desired speed (as defined by the external network) is not achievable with commodity hardware, parallelism is used to multiply the maximum data rate. Using the NSCP/SP2 configurations at Penn as an illustration, a group of 4 processors can be fed by a 100BaseT switch and four Fast Ethernet interfaces or (or, alternatively, by 4 OC3 ATM interfaces and an ATM switch). The OC3 ATM capability (nominally 155 Mbs) is degraded by ATM overhead and system performance, but 100Mbs per processor is achievable with either ethernet or ATM. For four processors, the resulting 400Mbit/s is roughly 50Mbytes/second. Fifty Mbytes(MB) per second from the four processors can be fed to parallel disks that are connected using Serial Storage Architecture with a transfer capability of 80MB/s. Interprocessor communication on the SP2 switches can sustain 90MB/s. Each SSA loop feeds a single processor, but that loop must contain a sufficiently large number of disks (typically ten) to saturate the loop, as the maximum data transfer rate from a single disk is about 8MB/s. The result is a system with several high-speed networks , and where the maximum rate at each stage of the data transfer from disk to network is well matched. This configuration for the parallel data system is scalable by replicating groups of processors.[18] It is shown schematically below.



Terabyte and larger data collections are becoming more common, but we do not have the technology today to *casually and quickly* explore remote data nor to mine data of this scale. Over the next several years, as the amount of dark fiber grows, bandwidth is expected to become a commodity. In this environment, it is possible to imagine a distributed instrument that allows the casual exploration and distributed mining of very large data sets.

On the other hand, even with OC-3 links, we found in NSCP that in practice only relatively small amounts of remote data (< 100 GB) could be moved or explored in a casual fashion. At its theoretical limit, an OC-3 can move a Terabyte of data in about 14 hours and a Gigabyte in about a minute. The NSCP-3 Meta-Cluster would be designed to facilitate the casual remote exploration of even larger data sets and the distributed mining of geographically distributed data. We take the viewpoint that the goal of a distributed computation is to complete the computation with sufficient accuracy while moving as little data as possible. Given data sets of the size under discussion, many computations require repetitive movement of 10 GB or more of data. This requires 8.6, 2.1 and 0.5 minutes using an OC-3, OC-12 and OC-48 respectively at its theoretical limit.

Much higher bandwidths are available internally within the local clusters, thus after taking advantage of parallel I/O within the cluster, a key requirement is to parallelize the data flow between the distributed components and to carefully match parallel storage to parallel CPU to parallel network paths. Below we explore these requirements. Such a system may require a dedicated hardware system for each local cluster whose function is to matching system internal I/O to external I/O and to maintain parallel streams on the network.

One could argue that a simpler non-parallel high-speed serial network connection would alleviate this problem, and certainly there is continuous progress in designing such networks. However, there is some evidence[19] that parallel transfer techniques from an application's point of view, are particularly well suited to long haul networks depending on the protocol and it's implementation (eg. TCP/IP). Therefore, we would argue that there is the potential for interesting new capabilities exploiting parallel links in a WAN environment that we can explore. Further, we expect such parallel links to appear soon in high performance metropolitan area networks and aggregation points and eventually on wide area networks

through the use of wave division multiplexing on fiber networks. Effective use of such a network requires careful integration of the wide area network and local networks into the cluster as proposed in this work.

Distributed data enabled by *parallel* network interconnects provides a number of interesting opportunities including much larger scalable bandwidth, the possibility of explicitly using the parallel streams for data striping, or using wavelengths for separated functions (e.g. cluster manager, network management, process migration, and data transfer). All would be useful for data intensive applications.

Today we are seeing a rapid growth of optical networks on the wide area and the beginnings of a rapid expansion of local and metropolitan area optical characteristics.[20]. Historically, their very high speed and quality has led to their preferred use on long lines for WAN backbones. Wave Division Multiplexing (WDM) is now being used to greatly increase the available bandwidth on these backbones by packing many wavelengths carrying separated data streams onto a single fiber. Until recently, these techniques were too expensive to consider for networking components in our designs, but this has changed with rapid advances in communications markets and hardware, and the reduced cost of this type of equipment.

Video data is likely to play an interesting role in the development of such systems. The real time characteristics of video and the necessity of delivering frames in a uniform stream are related to similar requirements in scalable parallel network designs. Video server applications include video on demand, distance learning, and interactive TV and thus are under rapid development in industry. By adapting some of this software and hardware technology for the transmission of data, we can expect improvements in overall transmission rates, file system and I/O efficiency. To set the scale, a single video stream is 1.5 to 6 Mbit/s, so a large 1000 stream server is comparable to OC48.

Stream starvation in such a server leads to poor file server performance. In video servers, poor performance due to stream starvation is immediately noticeable at the client end as poor video quality. Thus video systems and video server software have often paid particular attention to these issues. Stream starvation can be avoided by buffering data or (inefficiently) designing the server to have much more than the minimum required capacity. Video servers can also incorporate a request control mechanism to prevent overloading and a scheduling mechanism to assure stream delivery. Though the data content is different, many of the ideas used in video servers are also useful in designing data pumps whose goal is to supply a continuous stream of data to a network or remote application.

NSCP has used a component of IBM parallel file systems called Virtual Shared Disk[21] (VSD) in implementing parallel access to files for a medical storage project described in section 6. VSD is high speed and has the ability to support parallel access to files from a single application or from a number of applications running in parallel. We have also used IBM Videocharger software to build a video server and library. The underlying technology in the video server is called Tiger Shark[22] and takes advantage of real-time features of the AIX operating system including automatic balancing of loads across disks. Under development by IBM is integration of the Videocharger and Digital Library system with General Parallel File System (GPFS) which is based on VSD. Thus, we are beginning to see a merging of parallel high performance file systems and video technologies.

Returning to Lesson 1, the primary message is that rapid manipulation of large data collections requires parallel implementation wherever possible, and that extending and matching the parallelism between all of the components of the system, namely disk, CPU and network is required.

5. Lesson 2: Exploit Data Parallelism

Consider the problem of building a classifier on a large data set, say a terabyte of data. There are three fundamentally different approaches – sampling, task parallelism and data parallelism:

- *Sample, classify, and repeat*. With this approach, one samples the data to fit into memory, builds a classifier and validates it on another sample. This process is repeated until a satisfactory classifier is obtained or until 5pm, which ever comes earlier. With

this approach megabyte samples are extracted from the terabyte and most of the data is ignored. This is the traditional approach and is usually justified by the observation that memories are getting larger all the time.

- *Assign processors to those parts of computation that can be executed in parallel.* For example, when building a tree-based classifier, the most expensive part of the computation is the sorting and the computation of the splitting value. This can be easily done in parallel for nodes at the same level. As usual, there can be communication bottlenecks though as the algorithm proceeds. We note that some algorithms for computing tree-based classifiers exploit data structures so that the data can be pre-sorted once, instead of at each node. With is this approach, all the data is examined but much of it must be moved to those processors which are free.
- *Partition, classify and build an ensemble*. With this approach, one partitions the data, grows a classifier on each partition, and assembles the classifiers to produce an ensemble of classifiers[23]. An ensemble of classifiers is collection of classifiers with a rule for combining them. As an example, classifiers can be combined via a majority vote. With this approach, all the data is examined and all of the data is left in place.

Often the simplest approach is the most scalable – generally a data parallel approach is simpler than a task parallel approach. In the NSCP, we have generally used the third approach. In the remainder of this section, we discuss this approach in more detail.

One of the reasons that data parallel approaches have not become more popular is that the resulting object is not a classifier itself but rather an ensemble of classifiers, and ensembles are not as well understand as a single classifier. From the appropriate viewpoint however, ensembles are very natural objects. For example, consider a terabyte of data managed by a 100 node cluster, with each node containing 10 Gigabytes of local data, with the problem being to find a tree-based classifier for detecting transaction fraud. It is natural for each node independently and in parallel to construct a classification tree and then to ship the tree to a coordinating node which assembles the trees into a ensemble of trees. A transaction is evaluated for fraud by evaluating it individually on each of the 100 trees in the ensemble. Each tree individually indicates whether the transaction is fraudulent or not. A simple majority vote is then used to produce the final classification for the ensemble. This is a simple, but effective approach for working with Terabyte size data sets[24]

Another advantage of data parallel approaches involving ensembles is that this approach also works naturally for building classifiers on geographically distributed data. For example, in the NSCP, we have used ensembles of tree-based classifiers to analyze health care outcome data which is geographically distributed by the hospitals which produced it.

6. Lesson 3: Pay Attention to Physical Data Layout and Data Transformations.

Legacy Formats and Data Layout

Many projects with large data collections employ some form of database for maintaining that information. Along with the database come legacy data formats in which the data is held, standard reports, and probably considerable numbers of tools and software that accept the particular legacy format as input. Despite the comments made in the previous section about extracting simplified forms of the data, a problem occurs when the data collection size reaches the Terabyte level. Implementing Terabyte database systems is difficult, and copying or transforming them is time-consuming. Further, there may not be sufficient facilities to simultaneously maintain both a legacy format and a newer but more efficient

format. Thus, there has to be a tradeoff between a new form of database optimized for the data mining process, and using the data in its legacy form as much as possible.

An attractive approach is to design tools that interface the legacy format as much as possible to the data mining process. With the latter approach, the possibility of using the library of software and tools that rely on the legacy form during the mining process remains open. Furthermore, this approach eliminates the need to synchronize the mined data with another perhaps dynamic legacy database and reduces the total volume of stored information since one does not have to have two copies of the data, one in each format. The disadvantage is of course in efficiency of the mining process itself this is counterbalanced by the efficiency of legacy code reuse. Since high performance means getting accurate information as quickly as possible, sticking with the legacy form is an option that should not be discarded lightly.
A hybrid approach, which we have used in NSCP (http://nscp.upenn.edu), is to use index data sets to point back to the legacy format. In this approach, the legacy format is actually rearranged to optimize it for the mining process, but an index table allows the reconstruction of the original form. This optimizes the performance of the mining process at the slight expense of the other processes. New code for data transforms to legacy formats are required.

I/O efficiency through Data Layout

The key point in worrying about the data format is that its layout on the hardware seriously affects the performance of any system which needs to repetitively transfer the data from disk or other storage medium to CPU. As a simple example, consider a system that accumulates transaction records and places them in storage as they are accumulated. The data are time ordered, and this is convenient for answering questions about the time dependence of the data. For example, one can efficiently look up all transactions that occurred last February: Simply extract that portion of the data. Since these records are all adjacent and sorted, this can be done efficiently by the hardware. Suppose, however, that the subject of the data mining query is to know the zipcodes of everyone who had bought a particular product. This would require reading all of the data, making a particular selection (product name) and then accumulating the zipcodes. From a hardware point of view this could be very inefficient and could require reading a complete buffer to extract only a few words. If the database were resorted by product, or if we explicitly produced such a table, we could recover some of this efficiency. This same sitiuation occurs for example in re-sorting astronomical sky images that are collected and stored over time or for high energy particle events that are stored in time sequence as they are collected.

Many database products will handle some of these operations automatically, provided the database does not get too large to fit in memory. Some in fact automatically index frequently used quantities and store hashed index tables in the database or in memory. A more systematic approach is required for very large databases or distributed databases.

We conclude this section with two examples taken from NSCP applications[25] of large scale data using different indexing schemes both scalable and extendible to distributed systems. The first is take from the field of high energy particle physics which accumulates large samples of processed data in several year runs, accumulating 10-100 Terabytes of legacy format data per year. In this case, data transforms were used to access reformatted legacy data. The second comes from storage of digital radiology information. This is an extremely large collection that grows each year at a projected rate five years from now that could exceed 28 Petabytes per year. In this case, index information is extracted as data is collected leading to large meta-data collections.

Particle Physics detectors have hundreds of thousands of high-speed electronics elements that record the information about collision events occurring with MHz frequency. Therefore, particle detector experiments collect data at rates that can easily result in samples of 10-100 terabytes. Data samples for a CDF experiment at the Fermi National Accelerator Center including raw data, simulated data, and analysis data sets total approximately 100 terabytes. Future accelerators and detectors as well as upgrades

to existing detectors will increase this number to the petabyte range and above. Analysis of the data consists of statistical analysis of selected items, extensive analysis of correlations within records often with non-linear combinations of items, and significant amounts of visualization and CPU-intensive Monte Carlo studies.

The traditional data organization in particle physics uses sequential files with fixed-length blocking of variable length records. Data for an entire event is read, and selections of interesting events are made based on typically a small fraction of the total data. It has become common practice for many users to make a single pass through the data to collect summary information, recording the few hundred words per 200,000 bytes that are most useful. While this approach does speed data processing considerably, it requires the user to be very careful in the selection of the initial summary information – it is basically a survival mechanism in the face of a data flood. Other techniques including elimination of some data (data summary tapes or DST's), event selection, and summarization of data, all tend to limit the ability to use any analysis code that relies on full access to the complete data record.

Our strategy has been to rewrite the data in a form that makes fixed-length efficient parallel file access possible, while reducing the total amount of data read by a typical analysis program. At the same time, we try to maintain the ability to deal with the data in its original legacy form through data transforms. This has been accomplished by making a new structure for the input data format called "multifile".

The major advantage of the new form was that the data was broken down into commonly used components or objects, making it no longer necessary to read an entire event to begin processing. Instead, only those parts of the event necessary for a particular analysis need be read. The details of the format were arranged to lend themselves to parallel usage either by striping objects across disks or by distributing portions of collections of events across disks. This reorganization of the serial data within a record into a "multifile" structure collects specific types of data across events. Cross-reference tables connect individual data types or objects to companion information from the rest of the event record. Using this scheme, many elements of the event records become fixed length objects thus making parallel I/O easier. Elements that are variable in length within a single event because they contain a variable number of fixed-length elements also become fixed length. The major gain in this technique comes from being able to access only the segments of data that are needed or used in a particular analysis while preserving the remainder of the data for other users and in improving I/O efficiency through fixed blocking. The resulting collection was also implemented in a persistent store making the very large collection appear to be memory-resident using very lightweight database protocols.

Digitized radiology is becoming increasingly common as film methods are replaced by digital X-rays, and increasingly, medical diagnosis relies on intrinsically digital technologies (MRI, CAT, PET). Several archive strategies are being considered, but there is a considerable difference between archival storage and a database that holds the information in a form where it becomes useful as a database. NSCP has prototyped some elements of this problem in order to investigate the issues involved at the hardware and software level, and these designs are now the basis of a Next Generation Internet project being carried out jointly with four hospitals (http://nscp.upenn.edu/ndma). Our goal was to design a database/archive that could meet the needs of a radiology department, but that would also become a research tool capable of treating the aggregating information as a database. In principle the idea is very simple. What makes this particular task different from the traditional database problem is the size and complexity of the underlying data and that eventually it must be implemented for distributed systems.

A major radiology department can collect more than 6TB of data per year and there are approximately 2000 such units in the US alone, thus one can rapidly accumulate useful data at a scale which exceeds anything we have encountered before. These data have to be stored, manipulated, and retrieved fast enough so that doctors can access a record (image) in a few seconds or minutes at the most. Ideally, the storage system should also

be able to scan across the entire contents of a collection for records of interest or even for features embedded in the images.

We have addressed several aspects of this problem in a prototype system called RadAR (radiology archive). The choice of the hardware and the software design was based on the need to have a reliable and fast archive/database consistent with execution of research query functions. Initial tests have shown that the system would be able to easily accumulate 100GB per day per data engine and have remaining capacity for individual queries or for data mining since the internal bandwidth is much higher than required for handling just the input. The system has a very large front-end capacity, and we are confident that the system can scale well and perform as a simultaneous archive and data mining and querying engine. Further tests at the terabyte scale with additional data are the subject of the National Library of Medicine pilot.

In RadAR, data are received from the hospital in a legacy format called DICOM. A multithreaded process running on the input stream extracts the image portion of the record and additional attributes that are likely targets of a later query. From then on, portions of the records are stored in different parts of memory, disk, or tape resident databases, and a meta-database is kept with the location of all attributes of the stored DICOM record. This is similar to the multi-file approach. We have divided the storage space in three major areas. The short-term memory (disk resident) holds only one day's data and is the fastest access medium. The medium term storage is a parallel collection of fast hard disks. This area holds 3 months of data. Finally the long-term memory is a tape library. The long term storage is however not time ordered, but is sorted. Recently accessed records migrate to short term storage. All processes involved in receiving, migrating, searching, and retrieving are controlled by RadAR. RadAR is a parallel, multi-threaded; object oriented "data server". The front end is web-based. Web techniques will allow an unsophisticated user to execute parallel queries automatically and easily, either through simple forms or through SQL queries. Additional web controlled (java) processes direct input data to the database and monitor the database activity. Crucial data is mirrored and/or replicated on two systems. The result in this example is a system that simultaneously will store incoming data, construct ongoing meta-data catalogs, and enable use of the sorted collection as a database.

7. Conclusion

We have focused our attention on the lessons to be learned from attempts to manage massive distributed data. The only way to get fast enough performance for very large collections of externally stored information is to implement hardware-level parallelism on as many subsystems as possible. This means that the data should be distributed across parallel disk systems, the database should be parallel-enabled, and the CPUs should be a parallel system with high interprocessor bandwidth. Massive data is intrinsically distributed. It is managed and produced by distributed sensors and systems. Distributed management must play a key role, and when available, distributed systems offer an additional layer of parallelism at the meta-cluster level for parallel storage and analysis. Database systems must assist the data mining process by intelligently storing meta-information and taking advantage of data parallelism. Finally, a key to efficient use of the resulting parallel systems and data parallel analysis is the careful attention to data layout and indexing.

References:

[1] R. L. Grossman, S. Kasif, D. Mon, A. Ramu and B. Malhi, The Preliminary Design of Papyrus: A System for High Performance, Distributed Data Mining over Clusters, Meta-Clusters and Super-Clusters, KDD-98 Workshop on Distributed Data Mining, AAAI, 1999 and R. L. Grossman, S. Bailey, A. Ramu, B. Malhi and H. Sivakumar, A. Turinsky, Papyrus: A System for Data Mining over Local and Wide Area Clusters and Super-Clusters, Proceedings of Supercomputing 1999, IEEE.

[2] Thomas E. Anderson, David E. Culler, David A. Patterson, and the NOW Team, A Case for Networks of Workstations: NOW, IEEE Micro, Feb, 1995.

[3] D. J. Becker, T. Sterling, D. Savarese, E. Dorband, U. A. Ranawake , and C. V. Packer; BEOWULF: A Parallel Workstation for Scientific Computation, Proceedings of the 1995 International Conference on Parallel Processing (ICPP), 11-14, 1995.

[4] Yutaka Ishikawa, Mitsuhisa Sato, TomoHiro Kudoh, and Jyunichi Shimada. Towards a Seamless Parallel Computing System on Distributed Environments. In CPSY. SWOPP'97, August 1997

[5] I. Foster and C. Kesselman, Globus Project: A Status Report, Proceedings of the Heterogeneous Computing Workshop, IEEE, 1998, pages 4-18.

Andrew S. Grimsaw and William A. Wulf, Legion - A View From 50,000 Feet, Proceedings of the Fifth IEEE International Symposium on High Performance Distributed Computing, IEEE Computer Society Press, Los Alamitos, California, August 1996.

[6] S. Stolfo, A. L. Prodromidis, and P. K. Chan, JAM: Java Agents for Meta-Learning over Distributed Databases, Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, AAAI Press, Menlo Park, California, 1997.

[7] Y. Guo, S. M. Rueger, J. Sutiwaraphun, J.; and J. Forbes-Millott, Meta-Learnig for Parallel Data Mining, in Proceedings of the Seventh Parallel Computing Workshop, pages 1-2, 1997.

[8] H. Kargupta, I. Hamzaoglu and B. Stafford, Scalable, Distributed Data Mining Using an Agent Based Architecture, in D. Heckerman, H. Mannila, D. Pregibon, and R. Uthurusamy, editors, Proceedings the Third International Conference on the Knowledge Discovery and Data Mining, AAAI Press, Menlo Park, California, pages 211-214, 1997.

[9] T. G. Dietterich, Machine Learning Research: Four Current Directions, AI Magazine Volume 18, pages 97-136, 1997.

[10] A. E. Raftery, D. Madigan, and J. A. Hoeting, Bayesian Model Averaging for Linear Regression Models, Journal of the American Statistical Association Volume 92, 1996, pages 179-191.

[11] D. Wolpert, Stacked Generalization, Neural Networks Volume 5, 1992, pages 241-259.

[12] R. L. Grossman, H. Bodek, D. Northcutt, and H. V. Poor, "Data Mining and Tree-based Optimization," Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, E. Simoudis, J. Han and U. Fayyad, editors, AAAI Press, Menlo Park, California, 1996, pp 323-326.

[13] L. Xu, and M.I. Jordan, EM Learning on A Generalised Finite Mixture Model for Combining Multiple Classifiers, in Proceedings of World Congress on Neural Networks, Erlbaum, Hillsdale, NJ, 1993.

[14] R. Moore, T. A. Prince, and M. Ellisman, Data Intensive Computing and Digital Libraries, Communications of the ACM, Volume 41, pages 56-62, 1998.

[15] R. Subramonian and S. Parthasarathy, An Architecture for Distributed Data Mining, to appear.

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

R. L. Grossman, H. Bodek, D. Northcutt, and H. V. Poor, "Data Mining and Tree-based Optimization," Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, E. Simoudis, J. Han and U. Fayyad, editors, AAAI Press, Menlo Park, California, 1996, pp 323-326.

H.Karagupta, E. Johnson, E. R. Sanseverino, B-H Park, L. D. Silvestre, and D. Hershberger, Scalable Data Mining from Vertically Partitioned Feature Space Using Collective Mining and Gene Expression Based Genetic Algorithms, KDD-98 Workshop on Distributed Data Mining, to appear.

[17] R. L. Grossman, H. Bodek, D. Northcutt, and H. V. Poor, Data Mining and Tree-based Optimization, Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, E. Simoudis, J. Han and U. Fayyad, editors, AAAI Press, Menlo Park, California, 1996, pp 323-326.

[18] R. Hollebeek: “Data Intensive Computing, Data Mining, and Economic Development”, Proceedings of Supercomputing 1998, IEEE. http://nscp.upenn.edu/hollebeek/talks/sc98

[19] S. Bailey, R. Grossman, “Transport Layer Multiplexing with Psocket”, NCDM Technical Report, 1999.
[20] “Dark Fiber Economics”, David S. Isenberg, America’s Network, Dec. 1999.
[21] IBM Virtual Shared Disk Users Guide, Doc. No. GC23-3849-00, 1994.
[22] http://www.research.ibm.com/rd/422/haskin.txt
[23]R. L. Grossman, H. Bodek, D. Northcutt, and H. V. Poor, "Data Mining and Tree-based Optimization," Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, E. Simoudis, J. Han and U. Fayyad, editors, AAAI Press, Menlo Park, California, 1996, pp 323-326.
T. G. Dietterich, Machine Learning Research: Four Current Directions, AI Magazine Volume 18, pages 97-136, 1997.
[24] R. L. Grossman, H. Bodek, D. Northcutt, and H. V. Poor, Data Mining and Tree-based Optimization, Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, E. Simoudis, J. Han and U. Fayyad, editors, AAAI Press, Menlo Park, California, 1996, pp 323-326.
[25] R. Hollebeek: “Data Intensive Mining”, Proceedings of Supercomputing 1999, IEEE. http://nscp.upenn.edu/hollebeek/talks/sc99