A Requirements Analysis for Parallel KDD Systems

William A. Maniatty[1] and Mohammed J. Zaki[2]

[1] Computer Science Dept., University at Albany, Albany, NY 12222
maniatty@cs.albany.edu, http://www.cs.albany.edu/~maniatty/
[2] Computer Science Dept., Rensselaer Polytechnic Institute, Troy, NY 12180
zaki@cs.rpi.edu, http://www.cs.rpi.edu/~zaki/

Abstract. The current generation of data mining tools have limited capacity and performance, since these tools tend to be sequential. This paper explores a migration path out of this bottleneck by considering an integrated hardware and software approach to parallelize data mining. Our analysis shows that parallel data mining solutions require the following components: parallel data mining algorithms, parallel and distributed data bases, parallel file systems, parallel I/O, tertiary storage, management of online data, support for heterogeneous data representations, security, quality of service and pricing metrics. State of the art technology in these areas is surveyed with an eye towards an integration strategy leading to a complete solution.

1 Introduction

Knowledge discovery in databases (KDD) employs a variety of techniques, collectively called *data mining*, to uncover trends in large volumes of data. Many applications generate (or acquire) data faster than it can be analyzed using existing KDD tools, leading to perpetual data archival without retrieval or analysis. Furthermore, analyzing sufficiently large data sets can exceed the available computational resources of existing computers. In order to reverse the vicious cycle induced by these two problematic trends, the issues of performing KDD faster than the rate of arrival and increasing capacity must simultaneously be dealt with. Fortunately, novel applications of parallel computing techniques should assist in solving these large problems in a timely fashion.

Parallel KDD (PKDD) techniques are not currently that common, though recent algorithmic advances seek to address these problems (Freitas and Lavington 1998; Zaki 1999; Zaki and Ho 2000; Kargupta and Chan 2000). However, there has been no work in designing and implementing large-scale parallel KDD systems, which must not only support the mining algorithms, but also the entire KDD process, including the pre-processing and post-processing steps (in fact, it has been posited that around 80% of the KDD effort is spent in these steps, rather than mining). The picture gets even more complicated when one considers persistent data management of mined patterns and models.

Given the infancy of KDD in general, and PKDD in particular, it is not clear how or where to start, to realize the goal of building a PKDD system

that can handle terabyte-sized (or larger) central or distributed datasets. Part of the problem stems from the fact that PKDD draws input from diverse areas that have been traditionally studied in isolation. Typically, the KDD process is supported by a hierarchical architecture consisting of the following layers: (from bottom to top) I/O Support, File System, Data Base, Query Manager, and Data Mining. However, the current incarnations of this architecture tend to be sequential, limiting both problem size and performance. To implement a successful PKDD toolkit, we need to borrow, adapt, and enhance research in fields such as super-, meta- and heterogeneous-computing environments, parallel and distributed databases, parallel and distributed file systems, parallel I/O, mass storage systems, and so on (not to mention the other fields that make up KDD — statistics, machine learning, visualization, etc.).

This paper represents a first step in the process of unifying these diverse technologies and leveraging them within the PKDD system. We do this by discussing the system requirements for PKDD and the extant solutions (or lack thereof), i.e., the *what* and the *how* of PKDD. These requirements follow from: the basic requirements imposed by KDD (Section 2), current KDD algorithmic techniques (Section 3), the trends in commodity hardware design (Section 4) and software requirements (Section 5). One difficulty in making such a survey is that each research community has its own jargon, which we will try to make accessible by describing it within a common PKDD framework.

2 PKDD Requirements

We begin by discussing the wish-list or desirable features of a functional PKDD system, using it to guide the rest of the survey. We mainly concentrate on aspects that have not received wide attention as yet.

Algorithm Evaluation: Algorithmic aspects that need attention are the ability to handle high dimensional datasets, to support terabyte data-stores, to minimize number of data scans, etc. An even more important research area is to provide a rapid development framework to implement and conduct the performance evaluation of a number of competing parallel methods for a given mining task. Currently this is a very time-consuming process, and there are no guidelines when to use a particular algorithm over another.

Process Support: The toolkit should support all KDD steps, from pre-processing operations for like sampling, discretization, and feature subset selection, to post-processing operations like rule grouping and pruning and model scoring. Other aspects include (persistent) pattern management operations like caching, efficient retrieval, and meta-level mining.

Location Transparency: The PKDD system should be able to seamlessly access and mine datasets regardless of their location, be they centralized or distributed.

Data Type Transparency: The system should be able to cope with heterogeneity (e.g., different database schemas), without having to materialize a join of multiple tables. Other difficult aspects deal with handling unstructured (hyper-)text, spreadsheet, and a variety of other data types.

System Transparency: This refers to the fact that the PKDD system should be able to seamlessly access file systems, databases, or data archives. Databases and data warehouses represent one kind of data repositories, and thus it is crucial to integrate mining with DBMS to avoid extracting data to flat files. On the other hand, a huge amount of data remains outside databases in flat-files, web-logs, etc. The PKDD system must therefore bridge the gap that exists today between the database and file-systems worlds (Choudhary and Kotz 1996). This is required since database systems today offer little functionality to support mining applications (Agrawal *et al.* 1993), and most research on parallel file systems and parallel I/O has looked at scientific applications, while data mining operations have very different workload characteristics.

Security, QoS and Pricing: In an increasingly networked world one constantly needs access to proprietary third-party and other remote datasets. The two main issues that need attention here are security and Quality-of-Service (QoS) issues in data mining. We need to prevent unauthorized mining, and we need to provide cost-sensitive mining to guarantee a level of performance. These issues are paramount in web-mining for e-commerce.

Availability, Fault Tolerance and Mobility: Distributed and parallel systems have more points of failure than centralized systems. Furthermore temporary disconnections (which are frequent in mobile computing environments) and reconnections by users should be tolerated with a minimal penalty to the user. Many real world applications cannot tolerate outages, and in the presence of QoS guarantees and contracts outages, can breach the agreements between providers and users. Little work has been done to address this area as well.

In the discussion below, due to space constraints, we choose to concentrate only on the algorithmic and hardware trends, and system transparency issues (i.e., parallel I/O and parallel and distributed databases), while briefly touching on other aspects (a more detailed paper is forthcoming).

3 Mining Methods

Faster and scalable algorithms for mining will always be required. Parallel and distributed computing seems ideally placed to address these big data performance issues. However, achieving good performance on today's multiprocessor systems is a non-trivial task. The main challenges include synchronization and communication minimization, work-load balancing, finding good data layout and data decomposition, and disk I/O minimization.

The parallel design space spans a number of systems and algorithmic components such as the hardware platform (shared vs. distributed), kind of parallelism (task vs. data), load balancing strategy (static vs. dynamic), data layout (horizontal vs. vertical) and search procedure used (complete vs. greedy).

Recent algorithmic work has been very successful in showing the benefits of parallelism for many of the common data mining tasks including association rules (Agrawal and Shafer 1996; Cheung *et al.* 1996; Han *et al.* 1997; Zaki *et al.* 1997), sequential patterns (Shintani and Kitsuregawa 1998; Zak-

i 2000), classification (Shafer *et al.* 1996; Joshi *et al.* 1998; Zaki *et al.* 1999; Sreenivas *et al.* 1999), regression (Williams *et al.* 2000) and clustering (Judd *et al.* 1996; Dhillon and Modha 2000; S. Goil and Choudhary 1999).

The typical trend in parallel mining is to start with a sequential method and pose various parallel formulations, implement them, and conduct a performance evaluation. While this is very important, it is a very costly process. After all, the parallel design space is vast and results on the parallelization of one serial method may not be applicable to other methods. The result is that there is a proliferation of parallel algorithms without any standardized benchmarking to compare and provide guidelines on which methods work better under what circumstances. The problem becomes even worse when a new and improved serial algorithm is found, and one is forced to come up with new parallel formulations. Thus, it is crucial that the PKDD system support rapid development and testing of algorithms to facilitate algorithmic performance evaluation.

One recent effort in this direction is discussed by (Skillicorn 1999). He emphasizes the importance of and presents a set of cost measures that can be applied to parallel algorithms to predict their computation, data access, and communication performance. These measures make it possible to compare different parallel implementation strategies for data-mining techniques without benchmarking each one.

A different approach is to build a data mining kernel that supports common data mining operations, and is modular in design so that new algorithms or their "primitive" components can be easily added to increase functionality. An example is the MKS (Anand *et al.* 1997) kernel. Also, generic set-oriented primitive operations were proposed in (Freitas and Lavington 1998) for classification and clustering, which were integrated with a parallel DBMS.

4   Hardware Models and Trends

The current hardware trends are that memory and disk capacity are increasing at a much higher rate than their speed. Furthermore, CPU capacity is roughly obeying Moore's law, which predicts doubling performance approximately every 18 months. To combat bus and memory bandwidth limitations, caching is used to improve the mean access time, giving rise to Non-Uniform Memory Access architectures. To accelerate the rate of computation, modern machines frequently increase the number of processing elements in an architecture. Logically, the memory of such machines is kept consistent, giving rise to a shared memory model, called Symmetric Multiprocessing (SMP) in the architecture community and *shared everything* in the database community (DeWitt and Gray 1992; Valduriez 1993). However the scalability of such architectures is limited, so for higher degrees of parallelism, a cluster of SMP nodes is used. This model, called *shared-nothing* in database literature, is also the preferred architecture for parallel databases (DeWitt and Gray 1992).

Redundant arrays of independent (or inexpensive) disks (RAID) (Chen *et al.* 1994) has gained popularity to increase I/O bandwidth and storage capacity,

reduce latency, and (optionally) support fault tolerance. In many systems, since the amount of data exceeds that which can be stored on disk, *tertiary storage* is used, typically consisting of one or more removable media devices with a juke box to swap the loaded media.

In addition to the current trends, there have been other ideas to improve the memory and storage bottlenecks. *Active Disks* (Riedel *et al.* 1997) and *Intelligent Disks* (Keeton *et al.* 1998) have been proposed as a means to exploit the improved processor performance of embedded processors in disk controllers to allow more complex I/O operations and optimizations, while reducing the amount of traffic over a congested I/O bus. Intelligent RAM (IRAM) (Kozyrakis and Patterson 1998) seeks to integrate processing elements in the memory. Active disks and IRAM are not currently prevalent, as the required hardware and systems software are not commonly available.

5   Software Infrastructure

Since our goal is to use commodity hardware, much of the support for our desired functionality is pushed back into the software. In this section we discuss some of the system transparency issues in PKDD systems, i.e., support for seamless access to databases and file systems and parallel I/O. We review selected aspects of these areas.

The most common database constructions currently in use are relational databases, object oriented databases, and object-relational databases. The data base layer ensures referential integrity and provides support for queries and/or transactions on the data set (Oszu and Valduriez 1999). The data base layer is frequently accessed via a query language, such as SQL. We are primarily interested in parallel and distributed database systems (DeWitt and Gray 1992; Valduriez 1993), which have data sets spanning disks. The primary advantages of such systems are that capacity of storage is improved and that parallelizing of disk access improves bandwidth and (for large I/O's) can reduce latency. Early on parallel database research explored special-purpose database machines for performance (Hsiao 1983), but today the consensus is that its better to use available parallel platforms, with shared-nothing paradigm as the architecture of choice. Shared-nothing database systems include Teradata, Gamma (D. DeWitt et al. 1990), Tandem (Tandem Performance Group 1988), Bubba (Boral *et al.* 1990), Arbre (Lorie *et al.* 1989), etc. We refer the reader to (DeWitt and Gray 1992; Valduriez 1993; Khan *et al.* 1999) for excellent survey articles on parallel and distributed databases. Issues within parallel database research of relevance to PKDD include the data partitioning (over disks) methods used, such as simple *round-robin* partitioning, where records are distributed evenly among the disks. *Hash partitioning* is most effective for applications requiring associative access since records are partitioned based on a hash function. Finally, *range partitioning* clusters records with similar attributes together. Most parallel data mining work to-date has used a round-robin approach to data partitioning. Other methods might be more suitable. Exploration of efficient multidimensional indexing

structures for PKDD is required (Gaede and Gunther 1998). The vast amount of work on parallel relational query operators, particularly parallel join algorithms, is also of relevance (Pirahesh *et al.* 1990). The use of DBMS *views* (Oszu and Valduriez 1999) to restrict the access of a DBMS user to a subset of the data, can be used to provide security in KDD systems.

Parallel I/O and file systems techniques are geared to handling large data sets in a distributed memory environment, and appear to be a better fit than distributed file systems for managing the large data sets found in KDD applications. Parallel File Systems and Parallel I/O techniques have been widely studied; (Kotz) maintains an archive and bibliography, which has a nice reference guide (Stockinger 1998). Use of parallel I/O and file systems becomes necessary if RAID devices have insufficient capacity (due to scaling limitations) or contention for shared resources (e.g. buses or processors) exceeds the capacity of SMP architectures. The Scalable I/O initiative (SIO) includes many groups, including the *Message Passing Interface* (MPI) forum, which has adopted a MPI-IO API (Thakur *et al.* 1999) for parallel file management. MPI-IO is layered on top of local file systems. MPI uses a run time type definition scheme to define communication and I/O entity types. The ROMIO library (Thakur *et al.* 1999) implements MPI-IO in Argonne's MPICH implementation of MPI. ROMIO automates scheduling of aggregated I/O requests and uses the ADIO middleware layer to provide portability and isolate implementation dependent parts of MPIO. PABLO, another SIO member group, has created the *portable parallel file systems* (PPFS II), designed to support efficient access of large data sets in scientific applications with irregular access patterns. More information on parallel and distributed I/O and file systems appears in (Kotz ; Carretero *et al.* 1996; Gibson *et al.* 1999; Initiative ; Moyer and Sunderam 1994; Nieuwejaar and Kotz 1997; Schikuta *et al.* 1998; Seamons and Winslett 1996).

Users of PKDD systems are interested in maximizing performance. Prefetching is an important performance enhancing technique that can reduce the impact of latency by overlapping computation and I/O (Cortes 1999; Kimbrel *et al.* 1996; Patterson III 1997). In order for prefetching to be effective, the distributed system uses *hints* which indicate what data is likely to be used in the near future. Generation of accurate hints (not surprisingly) tends to be difficult since it relies on predicting a program's flow of control. Many hint generation techniques rely on traces of a program's I/O access patterns. (Kimbrel *et al.* 1996) surveyed a range of trace driven techniques and prefetching strategies, and provided performance comparisons. (Madhyastha and Reed 1997) recently used machine learning tools to analyze I/O traces from the PPFS, relying on artificial neural networks for on-line analysis of the current trace, and hidden markov models to analyze data obtained by profiling. (Chang and Gibson 1999) developed *SpecHint* which generates hints via speculative execution. We conjecture that PKDD techniques can be used to identify reference patterns, to provide hint generation and to address open performance analysis issues (Reed *et al.* 1998).

As we noted earlier, integration of various systems components for effective KDD is lagging. The current state of KDD tools can accurately be captured by

the term *flat-file mining*, i.e., prior to mining, all the data is extracted into a flat-file, which is then used for mining, effectively bypassing all database functionality. This is mainly because traditional databases are ill-equipped to handle/optimize the complex query structure of mining methods. However, recent work has recognized the need for integrating of the database, query management and data mining layers (Agrawal and Shim 1996; Sarawagi *et al.* 1998). (Agrawal and Shim 1996) postulated that better integration of the query manager, database and data mining layers would provide a speedup. (Sarawagi *et al.* 1998) confirmed that performance improvements could be attained, with the best performance obtained in *cache-mine* which caches and mines the query results on a local disk. SQL-like operators for mining association rules have also been developed (Meo *et al.* 1996). Further, proposals for data mining query language (Han *et al.* 1996; Imielinski and Mannila 1996; Imielinski *et al.* 1996; Siebes 1995) have emerged. We note that most of this work is targeted for serial environments. PKDD efforts will benefit from this research, but the optimization problems will of course be different in a parallel setting. Some exceptions include the parallel generic primitives proposed in (Freitas and Lavington 1998), and Data Surveyor (Holsheimer *et al.* 1996), a mining tool that uses the Monet database server for parallel classification rule induction. We further argue that we need a wider integration of parallel and distributed databases and file systems, to fully mine all available data (only a modest fraction of which actually resides in databases). Integration of PKDD and parallel file systems should enhance performance by improving hint generation in prefetching. Integrated PKDD can use parallel file systems for storing and managing large data sets and use distributed file system as an access point suited to mobile clients for management of query results.

6   Conclusions

We described a list of desirable design features of parallel KDD systems. These requirements motivated a brief survey of existing algorithmic and systems support for building such large-scale mining tools. We focused on the state-of-the-art in databases, and parallel I/O techniques. We observe that implementing a effective PKDD system requires integration of these diverse sub-fields into a coherent and seamless system. Emerging issues in PKDD include benchmarking, security, availability, mobility and QoS, motivating fresh research in these disciplines. Finally, PKDD approaches may be used as a tool in these areas (e.g. hint generation for prefetching in parallel I/O), resulting in a bootstrapping approach to software development.

References

R. Agrawal and J. Shafer. Parallel mining of association rules. *IEEE Trans. on Knowledge and Data Engg.*, 8(6):962–969, December 1996.

R. Agrawal and K. Shim. Developing tightly-coupled data mining applications on a relational DBMS. In *Int'l Conf. on Knowledge Discovery and Data Mining*, 1996.

R. Agrawal, T. Imielinski, and A. Swami. Database mining: A performance perspective. *IEEE Trans. on Knowledge and Data Engg.*, 5(6):914–925, December 1993.

S. Anand, *et al.* Designing a kernel for data mining. *IEEE Expert*, pages 65–74, March 1997.

H. Boral, *et al.* Prototyping Bubba, a highly parallel database system. *IEEE Trans. on Knowledge and Data Engg.*, 2(1), March 1990.

J. Carretero, *et al.* ParFiSys: A parallel file system for MPP. *ACM Operating Systems Review*, 30(2):74–80, 1996.

F. Chang and G. Gibson. Automatic i/o hint generation through speculative execution. In *Symp. on Operating Systems Design and Implementation*, February 1999.

P. M. Chen, *et al.* RAID: High-performance, reliable secondary storage. *ACM Computing Surveys*, 26(2):145–185, June 1994.

D. Cheung, *et al.* A fast distributed algorithm for mining association rules. In *4th Int'l Conf. Parallel and Distributed Info. Systems*, December 1996.

A. Choudhary and D. Kotz. Large-scale file systems with the flexibility of databases. *ACM Computing Surveys*, 28A(4), December 1996.

T. Cortes. *High Performance Cluster Computing*, Vol. 1, chapter Software Raid and Parallel File Systems, pages 463–495. Prentice Hall, 1999.

D. DeWitt et al. The GAMMA database machine project. *IEEE Trans. on Knowledge and Data Engg.*, 2(1):44–62, March 1990.

D. DeWitt and J. Gray. Parallel database systems: The future of high-performance database systems. *Communications of the ACM*, 35(6):85–98, June 1992.

I. S. Dhillon and D. S. Modha. A clustering algorithm on distributed memory machines. In *Zaki and Ho, 2000*.

A. Freitas and S. Lavington. *Mining very large databases with parallel processing*. Kluwer Academic Pub., 1998.

V. Gaede and O. Gunther. Multidimensional access methods. *ACM Computing Surveys*, 30(2):170–231, 1998.

G. Gibson, *et al.* NASD scalable storage systems. In *USENIX99, Extreme Linux Workshop*, June 1999.

J. Han, *et al.* DMQL: A data mining query language for relational databases. In *SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, June 1996.

E-H. Han, G. Karypis, and V. Kumar. Scalable parallel data mining for association rules. In *ACM SIGMOD Conf. Management of Data*, May 1997.

M. Holsheimer, M. L. Kersten, and A. Siebes. Data surveyor: Searching the nuggets in parallel. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*. AAAI Press, 1996.

D. Hsiao. *Advanced Database Machine Architectures*. Prentice Hall, 1983.

T. Imielinski and H. Mannila. A database perspective on knowledge discovery. *Communications of the ACM*, 39(11), November 1996.

T. Imielinski, A. Virmani, and A. Abdulghani. DATAMINE: Application programming interface and query language for database mining. In *Int'l Conf. Knowledge Discovery and Data Mining*, August 1996.

Scalable I/O Initiative. *http://www.cacr.caltech.edu/SIO.* California Institute of Technology.

M. Joshi, G. Karypis, and V. Kumar. ScalParC: A scalable and parallel classification algorithm for mining large datasets. In *Int'l Parallel Processing Symposium*, 1998.

D. Judd, P. McKinley, and A. Jain. Large-scale parallel data clustering. In *Int'l Conf. Pattern Recognition*, 1996.

H. Kargupta and P. Chan, editors. *Advances in Distributed Data Mining*. AAAI Press, 2000.

K. Keeton, D. Patterson, and J.M. Hellerstein. The case for intelligent disks. *SIGMOD Record*, 27(3):42–52, September 1998.

M.F. Khan, *et al.* Intensive data management in parallel systems: A survey. *Distributed and Parallel Databases*, 7:383–414, 1999.

T. Kimbrel, *et al.* A trace-driven comparison of algorithms for parallel prefetching and caching. In *USENIX Symp. on Operating Systems Design and Implementation*, pages 19–34, October 1996.

D. Kotz. The parallel i/o archive. Includes pointers to his Parallel I/O Bibliography, can be found at *http://www.cs.dartmouth.edu/pario/*.

C. E. Kozyrakis and D. A. Patterson. New direction in computer architecture research. *IEEE Computer*, pages 24–32, November 1998.

R. Lorie, *et al.* Adding inter-transaction parallelism to existing DBMS: Early experience. *IEEE Data Engineering Newsletter*, 12(1), March 1989.

T. M. Madhyastha and D. A. Reed. Exploiting global input/output access pattern classification. In *Proceedings of SC'97*, 1997. On CDROM.

R. Meo, G. Psaila, and S. Ceri. A new SQL-like operator for mining association rules. In *Int'l Conf. Very Large Databases*, 1996.

S. A. Moyer and V. S. Sunderam. PIOUS: a scalable parallel I/O system for distributed computing environments. In *Scalable High-Performance Computing Conf.*, 1994.

N. Nieuwejaar and D. Kotz. The galley parallel file system. *Parallel Computing*, 23(4), June 1997.

M. T. Oszu and P. Valduriez. *Principles of Distributed Database Systems*. Prentice Hall, 1999.

R. H. Patterson III. *Informed Prefetching and Caching*. PhD thesis, Carnegie Mellon University, December 1997.

Pirahesh *et al. Parallelism in Relational Data Base Systems*. In *nt'l Symp. on Parallel and Distributed Systems*, July 1990.

D. A. Reed, *et al.* Performance analysis of parallel systems: Approaches and open problems. In *Joint Symposium on Parallel Processing (JSPP)*, June 1998.

E. Riedel, G. A. Gibson, and C. Faloutsos. Active storage for large-scale data mining and multimedia. In *Int'l Conf. on Very Large Databases*, August 1997.

H. Nagesh S. Goil and A. Choudhary. MAFIA: Efficient and scalable subspace clustering for very large data sets. Technical Report 9906-010, Northwestern University, June 1999.

S. Sarawagi, S. Thomas, and R. Agrawal. Integrating association rule mining with databases: alternatives and implications. In *ACM SIGMOD Conf. on Management of Data*, June 1998.

E. Schikuta, T. Fuerle, and H. Wanek. ViPIOS: The vienna parallel input/output system. In *Euro-Par'98*, September 1998.

K. E. Seamons and M. Winslett. Multidimensional array I/O in Panda 1.0. *Journal of Supercomputing*, 10(2):191–211, 1996.

J. Shafer, R. Agrawal, and M. Mehta. Sprint: A scalable parallel classifier for data mining. In *Int'l Conf. on Very Large Databases*, March 1996.

T. Shintani and M. Kitsuregawa. Mining algorithms for sequential patterns in parallel: Hash based approach. In *2nd Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, April 1998.

A. Siebes. Foundations of an inductive query language. In *Int'l Conf. on Knowledge Discovery and Data Mining*, August 1995.

D. Skillicorn. Strategies for parallel data mining. *IEEE Concurrency*, 7(4):26–35, October-December 1999.

M. Sreenivas, K. Alsabti, and S. Ranka. Parallel out-of-core divide and conquer techniques with application to classification trees. In *Int'l Parallel Processing Symposium*, April 1999.

H. Stockinger. Dictionary on parallel input/output. Master's thesis, Dept. of Data Engineering, University of Vienna, February 1998.

Tandem Performance Group. A benchmark of non-stop SQL on the debit credit transaction. In *SIGMOD Conference*, June 1988.

R. Thakur, W. Gropp, and E. Lusk. On implementing mpi-io portably and with high performance. In *Workshop on I/O in Parallel and Distributed Systems*, May 1999.

P. Valduriez. Parallel database systems: Open problems and new issues. *Distributed and Parallel Databases*, 1:137–165, 1993.

G. Williams, *et al.* The integrated delivery of large-scale data mining: The ACSys data mining project. In *Zaki and Ho, 2000*.

M. J. Zaki and C-T. Ho, editors. *Large-Scale Parallel Data Mining*, LNCS Vol. 1759. Springer-Verlag, 2000.

M. J. Zaki, *et al.* Parallel algorithms for fast discovery of association rules. *Data Mining and Knowledge Discovery: An International Journal*, 1(4):343-373, December 1997.

M. J. Zaki, C.-T. Ho, and R. Agrawal. Parallel classification for data mining on shared-memory multiprocessors. In *Int'l Conf. on Data Engineering*, March 1999.

M. J. Zaki. Parallel and distributed association mining: A survey. *IEEE Concurrency*, 7(4):14–25, 1999.

M. J. Zaki. Parallel sequence mining on SMP machines. In *Zaki and Ho, 2000*.