ESPRIT Project No. 25 338

Work package G

Service Deployment

Survey on Monitoring Tools and Data Mining Techniques

ID:	DG1	Date:	17-Dec-97
Author(s):	Amsaleg, Billot, Le Nouy	Status:	
Reviewer(s):		Distribution:	

Change History

Document Code	Change Description	Author	Date
DG1	First version of document. No changes.	LA, MB, MLN	17-Dec-97

1. Introduction

This paper surveys the techniques that will be used as a basis for developing the software that fall into the workpackage-G, Service Deployment. We give here an overview of the basic techniques and issues related to performance monitoring and to data mining. Since our goal was not to be exhaustive, the interested reader may check the numerous references given at the end of this paper.

The structure of this survey is the following. Section 2 presents some fundamental notions related to performance monitoring. Some of the publicly available tools are also presented. Section 3 gives some basic knowledge about data mining techniques. Section 4 details how we intend to use performance monitoring and mining techniques to potentially improve the performance of the second pilot application, namely ETEL++.

2. Monitoring tools

In this document, we focus on monitoring tools allowing applications to take decisions related to load-balancing. These tools provide various numbers, each reflecting the usage of a resource like the CPU, the memory, the disk or the network links. Section 2.1 details some problems making performance monitoring a complex issue. Section 2.2 describes several monitoring tools that are commonly used under Unix and Windows. Finally, Section 2.3 presents a typical extension of these tools, namely a prediction feature that try to guess what could be the performance of the system in the short term, given the history of the evolution of performance in the past.

2.1. What to Monitor, How and When to Measure?

As soon as a computer is switched on, the operating system starts to monitor the resources it manages in order to deal with multiprogramming, sharing, memory allotment, scheduling, interrupts, etc... Resource monitoring is also possible for high-level applications because the operating system provides system calls that typically return elementary numbers reflecting the usage of a given resource. For example, Unix provides a handy call (`getrusage`) that returns information about resource utilization of the calling process (user and system times, numbers of input and output blocks, numbers of pages faults, ...).

The following resources are traditionally monitored: CPU load, disk utilization, memory consumption, I/O rate, caching hits, network packets loss, socket usage, etc...

On top of the basic calls provided by the operating system, tools or applications can build more sophisticated monitors to get indirectly numbers that reflect the utilization of a resource. A typical example is getting numbers reflecting the bandwidth of a network link. While no system calls directly deliver this information, the bandwidth can be for example deduced by observing the round trip time of a (say) 16K-bytes message. Note that the round trip time is obtained by invoking (at least twice) system calls returning the actual time of the day.

Regardless of getting performance directly or indirectly as mentioned above, monitoring tools are traditionally either exploiting immediately the numbers, or somehow compiling successive numbers in order to detect tendencies. The first approach is rather straightforward. The second approach raises the following issues.

In order to detect tendencies, monitoring tools have to get multiple and successive performance numbers. Deciding the appropriate frequency for sampling is rather difficult. Gathering numbers too frequently raises the risk of overloading the monitored resource with intrusive requests. A small sampling frequency is less disturbing, however, but raises the risk of missing major variations and thus the compilation of the gathered numbers can possibly be inaccurate. The frequency of sampling usually depends on the type of monitored resource, and is a trade-off between the additional load imposed over the resource and the confidence in the returned numbers. It also depends on the needs of the applications using the measurements. Typically, tools like `xload` or `taskmgr.exe` sample the system every second.

In addition to the frequency sampling issue, computing meaningful values based on the elementary numbers that are gathered is difficult. To isolate tendencies, to show the evolutions of resource usage, it is sometime necessary to filter-out transient bursts that may be irrelevant. This filtering is, as the frequency of sampling, a trade-off between being very reactive and returning numbers that show very short term phenomenon versus returning numbers that do not closely reflect the actual activity of the monitored resources but that may be more useful on the long term. Typically, filtering is based on the traditional notions of average, deviation, mean-square and smoothing. Another common type of filtering is based on a threshold: a number is returned only when the sample reflects an utilization that is above a given threshold. This typically requires an event manager or a notification mechanism.

Another issue -- of a higher level, however -- is the translation of the gathered numbers into a format that is useful for applications. Typically, tools transform the samples into arrays of numbers, histograms, graphs or pies.

2.2. A Short Overview of Existing Performance Monitoring Tools

This section presents several well-known tools that monitor systems, and focuses on tools that monitor the network. Their very goal is to deliver to humans numbers that reflect as closely as possible the activity of the monitored resources. None of these tools take decisions about what to do with the gathered data, that is, none try to balance the load of the system depending on the observed tendencies for example. Rather, these tools are “spies”, and they help users to take smart decisions about tuning their system by providing accurate, confident and fully trusted numbers. Although the focus of this section is network tools, we first, however, briefly mention typical tools that monitor machines.

2.2.1. Monitoring Machines

Every single machine provides low level system calls and some elementary tools of higher level that monitor the machine itself. Not surprisingly, monitoring is attached to the notion of process, that is, returned data shows how each process uses the resources. As the amount of information delivered for every process is rather large, it is sometimes better to merge process-based data and to extract from this merging higher level notions like the resource usage of a group of process, or process that are owned by the same person, or process having parent-child relationships, etc…

The tools generally available deliver roughly the same information, and only their interface with the user makes each specific. For information, we could mention `xload`, `sar`, `vmstat`, `taskmgr` and `top` as the tools that are the most frequently available on Unix and Windows based systems.

2.2.2. Monitoring the Network

Since mid-1995, many groups initiated extensive monitoring of the best known wide area network: the Internet. Their goal was to encourage research laboratories and companies around the globe to initiate rigorous joint studies in order to (i) define good metrics to evaluate the performance of the Internet and (ii) increase the knowledge level we have about the behavior of the network by performing extensive measurements on the short and long term. Several working groups tackled this problem: The IP Performance Metrics and the Realtime Traffic Flow Measurement groups of the Internet Engineering Task Force (IETF) (see http://www.ietf.org), the Regional Internet Registry for Europe (RIPE) (see http://www.ripe.net), and others.

In general, these groups have performed extensive measurements, and some pieces of the software they developed is now publicly available. We mention here two such tools that can be reused on a smaller-scale basis.

The Realtime Traffic Flow Measurement group of IETF made publicly available their software called NeTraMet (see http://www.auckland.ac.nz/net/Internet/rtfm). NeTraMet is an accounting meter which runs on a PC under DOS or a Unix system. It builds up packet and byte counts for traffic flows, which are defined by their end-point addresses. Addresses can be Ethernet addresses, protocol addresses (IP, DECnet, EtherTalk, IPX or CLNS, IP port numbers, etc), or any combination of these. The traffic flows to be observed are specified by a set of rules. Traffic flow data is collected via SNMP from NeTraMet by a specific program. NeTraMet provides a valuable tool for analyzing network traffic flows, and should prove to be of interest to anyone interested in network monitoring, capacity planning, performance measurement, etc.

Another very powerful tool is NNStat. NNStat has been originally developed in 1992 by R. Braden and A. DeSchon at the University of Southern California, Information Sciences Institute. NNStat is a collection of programs that provides an Internet statistic collecting capability. The NNStat strategy for statistic collection is to collect traffic statistics via a promiscuous Ethernet tap on the local networks, versus instrumenting the gateways. If all traffic entering or leaving a network or set of networks traverses a local Ethernet, then by stationing a statistic gathering agent on each local network a profile of network traffic can be gathered. Statistical data is retrieved from the local agents by a global manager. The NNStat distribution comes with several sample `awk` programs which process the logged output of the collect program.

Most of the detailed softwares are based on the following tools:

- Tools similar to the “Ping” call of Unix
- The protocols SNMP and CMIP

SNMP has been designed in the mid-1980's as an answer to the communication problems between different types of networks. The way it works is very simple: it exchanges network information through messages (called protocol data units). From a high-level perspective, each message is an object that contains variables that have both names and values. There are five types of messages that SNMP employs to monitor a network: two deal with reading terminal data, two deal with setting

terminal data, and one, the trap, is used for monitoring network events such as terminal start-ups or shut-downs. Each variable consists of the following information:

- the name of the variable
- the data type of the variable (e.g., integer, string)
- whether the variable is read-only or read-write
- the value of the variable

Basically, SNMP allows to monitor each site running an SNMP agent. The main advantage of this protocol is that it is in wide use. Therefore, there are a lot of already available agents which provide monitoring of network activity and hardware characteristics. Some of them also monitor the usage of memory and disks.

2.3. Predictions of Possible Behaviors

Monitoring tools typically accumulate the gathered data into profiles, and work on the profiles to deliver meaningful results to users. Some tools (e.g., NeTraMet) are also able to predict a possible behavior for the network based on all the gathered numbers that form somehow an history of past performance. Predicting a possible behavior for the performance is interesting to plan in advance, to anticipate, in order for example to enforce the quality of a service, or to issue bulk-transfers at a specific time.

3. Data-mining Techniques

The ever growing number of databases related applications (for the purpose of business, science and engineering for example) raised the need for new techniques and for new tools that can extract useful information from the stored data in order to provide valuable additional knowledge. This extraction, or discovery of knowledge in databases, is also called *data mining*. It corresponds to a non-trivial extraction of implicit, previously unknown and potentially useful information from data in databases [PSF91, ACF94, AMS96, Han96, FPSSU96]. By mining large databases, interesting knowledge, regularities, specific patterns and high-level information can be extracted and investigated from different angles. The discovered knowledge can be applied to information management, query processing, decision making, process control, artificial intelligence, statistics and data visualization. Furthermore, several emerging application for information providing services, such as on-line services and the World Wide Web, also call for various data mining techniques to better understand user behavior and to ameliorate the service provided. In general, data mining offers opportunities for major revenues.

Data mining poses many challenging issues. Due to space limitations, we unveil here only the tip of the iceberg: many other interesting issues are presented in [CHY96, CM96]. A first issue is related to the diversity of data types (typically ranging from relational data to complex objects) and to the different goals of mining. This diversity make it difficult for one mining system to handle all kinds of data. Rather, specific data mining systems are constructed, each offering dedicated solutions to knowledge mining. Another problem comes from the performance of mining algorithms. To effectively extract information from a huge amount of data, the mining algorithms must be efficient and scalable to very large data sets. This poses severe constraints on the complexity of the mining algorithms, and usually require elegant implementations. The usefulness, certainty, expressiveness and accuracy of the mined results is another source of hard problems. This becomes a major concern especially when the database used for mining is updated.

Data mining is an application-dependent issue, and different applications may require different mining techniques to cope with. In general, however, the kinds of knowledge which can be discovered fall in one of the following categories: mining association rules, multi-level data generalization, summarization and characterization, data classification, data clustering, pattern-based similarity search and mining sequential patterns. Each category is briefly detailed in the following sections.

3.1. Mining Association Rules

Typically, mining *association rules* means discovering in a database of sales transactions the important associations among items, such that the presence of some items in a transaction will imply the presence of other items in the same transaction. Two classical example are the discovery that ``80% of the customers purchasing milk also purchase bread" and that ``70% of the people buy wheat bread if they buy 2% milk". Such knowledge is useful for example to better organize the shelves, or to develop grouped promotions.

Mining association rules may require to repeatedly scan through the entire database to find different association patterns. As such, efficient algorithms and some methods for further performance enhancement are needed. [AIS93, AS94, PCY95] present typical algorithms that are efficient and also representative of mining association rules techniques.

Mining association rules can be expressed as follows. Let D be a set of transactions, such that each transaction contains a subset of items of the itemset I (e.g., I is the set of *all* the items you can purchase in your local store. A transaction is what you actually bought the other day). An association rule is an implication of the form $X \Rightarrow Y$, where $X \subseteq I$, $Y \subseteq I$ and $X \cap Y = \emptyset$. The rule $X \Rightarrow Y$ holds for the transaction set D with *confidence c* if c% of transactions in D that contain X also contain Y. The rule $X \Rightarrow Y$ has *support s* in the transaction set D if s% of transactions in D contains .

Confidence denotes the strength of implication and support indicates the frequency of the occurring patterns in rule. Only those rules that have a reasonably large support are considered.

The algorithms cited above typically implements the following two steps:

Discover the large itemsets L, i.e., the sets of itemsets that have a support above a pre-determined minimum support s.

For each large itemset $l \in L$, search for the association rules $l_1 \Rightarrow l_2$ with $l_1 \subseteq L$, $l_2 \subseteq L$, $l_1 \cap l_2 = \emptyset$ and $l_1 \cup l_2 = l$. Select the rules that have a confidence above a pre-determined minimum confidence c.

It is noted that the overall performance of mining association rules is determined by the first step -- deriving the rules from the itemsets being straightforward.

3.2. Multi-Level Data Generalization, Summarization and Characterization

Data often contain detailed information *at primitive concept levels*. It is often desirable to summarize a large set of data and present it at a high concept level. This functionality is achieved by the class of mining algorithms that are presented in this section. It is important to note that data generalization is the most popular way to use mining algorithms, often referred to as the *data cube approach* [Squ95, Moh 96, Inm 96, HRU 96, HAMS 97].

The general idea behind the data cube approach (also called OLAP, for On-Line Analytical Processing) is to materialize (i.e., pre-compute, typically off-line) and store certain *expensive* computations that are frequently inquired, especially those involving aggregate functions such as count, sum, aver-

age, max, etc. The materialized views are stored in a multi-dimensional database called a data cube (or a data warehouse) that can be interrogated for the purpose of knowledge discovery. Typically, values are grouped into a hierarchy. For example, the transactions reflecting the purchases of customers on a every day basis can be grouped into ``week", ``month" and ``year", and for each group, an aggregate value of the sales for that period is automatically computed. Generalizations and specializations can be performed on the data cube by roll-up and drill-down operations. The data cube approach is a valuable technique for many business-oriented applications, since it provides many different views on the same basic data, as it enforces performances by pre-computing frequently asked questions. [CHY 96] also details a technique for on-line data generalization, called the attribute-oriented induction approach.

3.3. Data Classification

Data classification is the process which finds the common properties among a set of objects in a database and classifies them into different classes. The data is classified according to a model, which is usually constructed from a sample extracted from the real database. For example, it might be desirable for a insurance company to classify its customers according to the date of their last crash, and to subsequently mine that such customers are likely to be single, male, below 25 years old, living in large cities. A well know approach to data classification is the use of decision trees [Qui86, MRA95, SAM96]. In this approach, a small sample is first used to built an initial decision tree. If the tree does not model accurately enough the complete set of data, a selection of the exceptions is added to the sample, and the process continues until the correct decision tree is found.

3.4. Data Clustering

The process of grouping physical or abstract objects into classes of similar objects is called clustering. Clustering analysis helps construct meaningful partitions of a large set of objects based on a divide-and-conquer approach which decomposes a large scale system into smaller components. In the context of data mining, data clustering identifies densely populated regions, according to some distance measurement, in a large, multidimensional data set [NH94, ZRL96]. In other words, data clustering tries to discover the overall distribution patterns of a data set, and thus facilitates taxonomy.

3.5. Pattern-Based Similarity Search

Temporal or spatial-temporal data constitutes a significant portion of the data stored in databases. Typical examples are financial databases for stock price index and medical databases. Searching for similar patterns in such databases is useful to discover and predict the risk, causality and trend associated with a specific pattern. Classical examples of queries for this type of database include identifying companies with similar growth patterns, products with similar selling patterns, stocks with similar price movements, tumors with similar initial characteristics, similar weather patterns, etc.

These queries invariably require similarity matches as opposed to exact matches [AFS93, FRM94, ALSS95].

3.6. Mining Sequential Patterns

Mining sequential patterns is somehow similar to the mining of association rules, but differs due to the additional notion of time embedded in the data and exploited by the mining algorithm. In this approach, mining algorithms not only search for large itemsets, but try to discover frequent *sequences* of items that are common to many transactions. This mining can for example reveal that a typical client successively purchase a tv-set, then a video recorder and finally a digital camera. To extract this knowledge, the mining algorithm had to investigate all the successive purchases of all the clients over the last two years.

An recent and interesting development of mining sequential patterns deals with the World Wide Web. In this environment, documents are usually linked together in some way to facilitate interactive access. Understanding user access patterns may help improving the design of the system (e.g., providing efficient access between highly correlated documents), and also allows better marketing decisions to be taken (e.g., putting advertisements in proper places). Capturing user access patterns in such environment is referred to as mining path traversal patterns. [CPY96] highlights a fundamental difference between mining path traversal and other kind of mining. In the context of path traversal, user are traveling along the information providing services to search for the desired information, and as such, some objects are visited because of their location rather than their content. This unique feature increases the difficulty of extracting meaningful information from a sequence of traversal.

4. Deployment of Etel++

In the context of FollowMe, we plan to use performance monitoring and algorithms that mine association rules to enhance the performance of the pilot application ETEL++. ETEL++ offers an online version of a regional newspaper. In addition to the traditional features of online services (like navigation, support of multimedia information, ...), ETEL++ will support personalization, i.e., ETEL++ will allow users to choose which areas the personalized newspaper will cover. The choices for a user are saved into his *user profile*.

Another feature of ETEL++ will be its ability to deliver data according to several different formats in order to support access from various types of terminals (e.g., PCs, portable phones, palmtops). Supporting many user-terminals (output devices) means that many different physical representations of the same logical data exist. All those representations find their origin in the same elementary structure (the real newspaper, directly out of the journalists hands), and differ in their media type. For example, an article represented originally as a text file can be converted into an audio file or an postcript file in order to be delivered respectively over a phone and on a laser printer. It is important to note here that the derived representations of a newspaper dramatically increase the volume of data to manage.

Because ETEL++ has to manage a large number of users, because it has to manage large volumes of data representing the newspaper and its derived representations, load-balancing policies appear to be a key point for the efficient deployment of ETEL++. In this section, we give an overview of the way we intend to use monitoring tools and mining techniques in this context.

4.1. Exploiting Performance Monitoring to Balance Computation and Communication Costs

From a bird’s eye-view, the architecture on top of which ETEL++ is built is made of three entities:

1) A central server located at Ouest-France (information provider). On this server are stored the articles once they are produced by the journalists. This server is connected to several secondary servers described next.

2) Secondary servers that are FollowMe-enabled machines. Each of these server has storage and computing power. To a secondary server are connected a set of user terminals described next.

3) User terminals that can be for example personal computers. A user ultimately browses its personalised electronic version of the newspaper on his terminal.

Allowing a user to browse his newspaper essentially consists in (i) determining what type of terminal is used, (ii) what are the chosen areas of information to cover and (iii) sending the relevant data from the central server to the user terminal. It is important to note here that the original (raw) data is created inside the Ouest-France server and that the derived data is displayed on users' terminal. As such, it is crucial to determine where that transformation takes place. Not surprisingly, three locations have to be considered:

- Inside Ouest-France's central server. In this case, all the possible versions are computed once, and the relevant versions can be either pushed or pulled by each secondary server before being delivered to users. The load on these secondary servers or on the terminal used by the users is almost zero: all the computations have been already done. The network, however, could become a bottleneck and might be severely congested. In addition, note that the Ouest-France machine is also heavily loaded.

- Inside each secondary server. With this approach, raw data is transmitted to these servers. Raw data tend to be of a much smaller size than the derive counterpart, and only a single version of that raw data travels around. In this case, all secondary servers have to compute their own derived versions. Two different servers will certainly make redundant computations since they are likely to generate at least a certain number of similar versions. With this approach, the burden is better distributed. Secondary servers, however, need a fairly reasonable amount of storage to keep the derived versions, and must also have a good processing power. This could not be the case for every single machine, consequently creating another set of problems.

- Inside the user's terminal. In this case, only the relevant version is computed using the raw data transmitted to the user terminal itself. In this case, the user terminal is loaded, and must be able to perform this computation efficiently. This approach might not be practical since user terminals are likely to have very different power and storage characteristics. Furthermore, storage space and computing power is somehow wasted since secondary servers are not involved.

It is possible to trade the costs of computing the derived versions against the costs of sending the versions via the network by appropriately choosing the location(s) of data processing. Computing the derived data very early (i.e., close to the central server) reduces computing costs (done once) while it increases the communication costs. The opposite applies if the derived data is computed very late, that is, close to the users. A flexible computing strategy would be to compute the derived versions using a carefully chosen combination of the previous locations. For example, the central server may compute a subset of the derived data, the rest may be divided among the secondary servers, and communications between servers could be enforced to leverage. In this case, the degree of computation redundancy, the amount of replication could be set such that it minimizes the overall cost.

Determining the best configuration is hard, and is certainly a NP-complete problem. Monitoring tools might be helpful in this case, because they can be used to define heuristics improving the behavior of the system. Observing the load of resources, and also being able to foresee a possible behavior of

those resources might help ETEL++ in finding a good placement of its computations and its data in order to maximize its efficiency.

4.2. Mining User Profile to Group Users on Servers

To enforce the quality of service offered by ETEL++, we try to build a load-balancing policy that is partly based on the mining of association rules between the profiles of users. By grouping users that have close profile in terms of their personal choices, it is possible to assign group of users to specific servers that would best serve their requests. Assigning groups to the relevant servers help to balance the load across all the available servers, each managing a reduced set of users.

We specifically intend to use mining algorithms to find the themes users are likely to access frequently (i.e., large itemsets using the data mining vocabulary). Given that knowledge, groups can be created and then assigned onto servers. While the use of mining technology is rather straightforward in our case, finding the best mapping in order to create groups that will improve significantly the performance is still an issue. Finding efficient solutions is part of our ongoing work.

5. References

ACF94

Rakesh Agrawal, Michael J. Carey, Christos Faloutsos, Sakti P. Ghosh, Maurice A. W. Houtsma, Tomasz Imielinski, Balakrishna R. Iyer, A. Mahboob, H. Miranda, Ramakrishnan Srikant, and Arun N. Swami. Quest: A project on database mining. In Richard T. Snodgrass and Marianne Winslett, editors, *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, page 514, Minneapolis, Minnesota, 24-27 May 1994.

AFS93

Rakesh Agrawal, Christos Faloutsos, and Arun Swami. Efficient similarity search in sequence databases. *Lecture Notes in Computer Science*, 730:69, 1993.

AIS93

Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207-216, Washington, D.C., 26-28 May 1993.

ALSS95

R. Agrawal, K.-I. Lin, H. S. Sawhney, and K. Shim. Fast similarity search in the presence of noise, scaling, and translation in time-series databases. In Umeshwar Dayal, Peter M. D. Gray, and Shojiro Nishio, editors, *VLDB '95: proceedings of the 21st International Conference on Very Large Data Bases, Zurich, Switzerland, Sept. 11-15, 1995*, pages 490-501, Los Altos, CA 94022, USA, 1995. Morgan Kaufmann Publishers.

AMS96

Rakesh Agrawal, Manish Mehta, John Shafer, Ramakrishnan Srikant, Andreas Arning, and Toni Bollinger. The quest data mining system. page 244. AAAI Press, 1996.

AS94

R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In Jorgeesh Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago, Chile proceedings*, pages 487-499, Los Altos, CA 94022, USA, 1994. Morgan Kaufmann Publishers.

CHY96

Ming-Syan Chen, Jiawei Han, and Philip S. Yu. Data mining: an overview from a database perspective. *Ieee Trans. On Knowledge And Data Engineering*, 8:866-883, December 1996.

CM96

Chris Clifton and Don Marks. Security and privacy implications of data mining. In *Workshop on Data Mining and Knowledge Discovery*, number 96-08, pages 15-19, Montreal, Canada, June 2 1996. ACM SIGMOD, University of British Columbia Department of Computer Science.

CPY96

M.-S. Chen, J. S. Park, and P. S. Yu. Data mining for path traversal patterns in a web environment. In *ICDCS '96; Proceedings of the 16th International Conference on Distributed Computing Systems; May 27-30, 1996, Hong Kong*, pages 385-393, Washington - Brussels - Tokyo, May 1996. IEEE.

FPSSU96

U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*. MII Press, Mento Park, 1996.

FRM94

C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 23(2):419-429, June 1994.

HAMS97

Ching-Tien Ho, Rakesh Agrawal, Nimrod Megiddo, and Ramakrishnan Srikant. Range queries in OLAP data cubes. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 26(2):73, 1997.

Han96

Jiawei Han. Data mining techniques. In H. V. Jagadish and Inderpal Singh Mumick, editors, *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, page 545, Montreal, Quebec, Canada, 4-6 June 1996.

HRU96

Venky Harinarayan, Anand Rajaraman, and Jeffrey D. Ullman. Implementing data cubes efficiently. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 25(2):205, 1996.

Inm96

W. H. Inmon. The data warehouse and data mining. *Communications of the ACM*, 39(11):49-50, November 1996.

Moh96

N. Mohan. DWMS: Data warehouse management system. In T. M. Vijayaraman et al., editors, *Proceedings of the twenty-second international Conference on Very Large Data Bases, September 3-6, 1996, Mumbai (Bombay), India*, pages 588, Los Altos, CA 94022, USA, 1996. Morgan Kaufmann Publishers.

MRA95

Manish Mehta, Jorma Rissanen, and Rakesh Agrawal. MDL-based decision tree pruning. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD'95)*, pages 216-221, August 1995.

CY95

Jong Soo Park, Ming-Syan Chen, and Philip S. Yu. An effective hash based algorithm for mining association rules. In Michael J. Carey and Donovan A. Schneider, editors, *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pages 175-186, San Jose, California, 22-25 May 1995.

PSF91

Gregory Piatetsky-Shapiro and William Frawley, editors. *Knowledge Discovery in Databases*. The MIT Press, Cambridge, MA, 1991.

Qui86

J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1:81-106, 1986.

SAM96

J. Shafer, R. Agrawal, and M. Mehta. SPRINT: A scalable parallel classifier for data mining. In T. M. Vijayaraman et al., editors, *Proceedings of the twenty-second international Conference on Very Large Data Bases, September 3-6, 1996, Mumbai (Bombay), India*, pages 544-555, Los Altos, CA 94022, USA, 1996. Morgan Kaufmann Publishers.

Squ95

Cass Squire. Data extraction and transformation for the data warehouse. In Michael J. Carey and Donovan A. Schneider, editors, *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pages 446-447, San Jose, California, 22-25 May 1995.

ZRL96

Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH: An efficient data clustering method for very large databases. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 25(2):103, 1996.