RESEARCH AND DEVELOPMENT
ERCIM News No.44 - January 2001 [contents]

Design of Efficient Input/Output Intensive Data Mining Applications

by Ranieri Baraglia, Domenico Laforenza, Salvatore Orlando, Paolo Palmerini and Raffaele Perego

---

The goal of Data Mining is to discover knowledge hidden in data repositories. This activity has recently attracted a lot of attention. Because of the huge datasets to be accessed, most data mining algorithms must consider the I/O actions carefully, hiding or minimizing their effects. At CNUCE-CNR we are studying how the architectural features of modern high performance computers and operating systems can be exploited when designing efficient data mining algorithms that can cope with huge datasets.

High energy physics experiments produce hundreds of TBytes of data, credit card companies hold large databases of customer's transactions, web search engines collect web documents worldwide. Regardless of the application field, Data Mining (DM) allows to 'dig' into huge datasets to reveal patterns and correlations useful for high level interpretation. Finding clusters, association rules, classes and time series are the most common DM tasks. All require the use of algorithms whose complexity, both in time and in space, grows at least linearly with the database size.

In recent years much R&D has been focused on the design of hardware and software systems that can cope with the growth in the dimensions of the data. Nevertheless, the amount of physical memory of modern computers is still in orders of magnitude lower than the size of many databases. Various strategies can be applied to address this problem. One approach investigates the development of new algorithms that reduce the need for data, either by exploiting sampling techniques or by limiting direct access to the database. These methods usually introduce some inaccuracy or useless CPU overhead. Another possibility is the exploitation of High Performance Computing Systems, which can extend the range of applicability of DM algorithms, without changing the conceptual limitation.

Many DM algorithms require a computation to be iteratively applied to all records of a dataset. In order to guarantee scalability, even on a serial or a small scale parallel platform (eg a workstation cluster), the increase in the I/O activity must be carefully taken into account. We have recognized two main categories of algorithms, with respect to the patterns of their I/O activities: Read and Compute (R&C) algorithms, which use the same dataset at each iteration, and Read, Compute and Write (RC&W) ones, which, at each iteration, rewrite the dataset to be used at the next step. A typical reason for writing a new dataset is to 'prune' the original one.

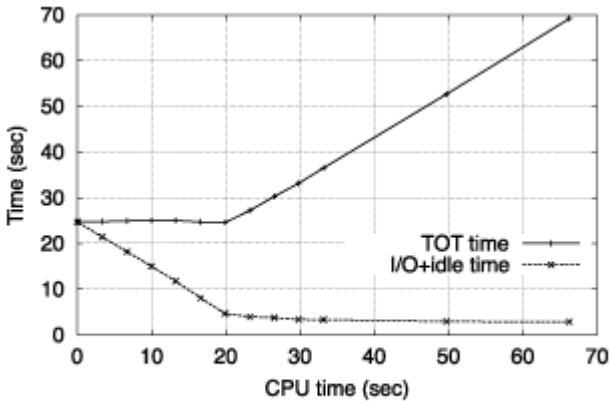
The first class includes Clustering algorithms, such as 'k-means', or Frequent Set Counting (FSC) algorithms, such as 'Apriori', while the second class covers some variants of 'Apriori', like the Direct Hash and Prune (DHP) algorithm, or the C4.5 algorithm for classification.

Modern operating systems (OS) provide effective mechanisms to access secondary storage units. While highly local problems can exploit OS caching and prefetching policies, the OS may fail to efficiently handle data movements through the levels of the

memory hierarchy. The employment of 'Out-of-Core' (OOC) techniques which explicitly take care of data movements is thus often a must for performance reasons. Another important OS feature is time-sharing among processes. Multi-threading can be used to overlap I/O actions with useful computations.

We have demonstrated that it is possible to take advantage of such features in order to design efficient OOC DM algorithms that hold all the data on disk, and that these algorithms can be effectively scaled by exploiting High Performance Computing Systems, such as a cluster of Symmetric Multiprocessors (SMPs). The main results are summarized in the following points:

- In R&C DM algorithms we find that OOC techniques perform better than the in-core counterparts. This is mainly because data locality can be exploited and I/O actions can be overlapped with computation. The figure refers to a synthetic kernel code inspired by this class of DM algorithms. In this kernel, we scan a file of 256 MB sequentially, reading blocks of 4 KB. We artificially varied the granularity of the computation performed on each block of data. When the total CPU time is greater than 20 s, the I/O and idle waiting times become negligible due to the prefetching OS activities. We discovered that most of the interesting DM problems belonging to the R&C class can be solved by algorithms characterized by high granularities; this means that our OOC techniques are able to completely hide I/O overheads.
- Multi-threading used as a general technique to hide I/O waiting time may conflict with prefetching, another technique independently adopted by the OS to reduce the impact of I/O when files are accessed sequentially. We developed techniques to take advantage of both multi-threading and prefetching when implementing OOC DM algorithms.
- On clusters of SMPs, data parallel approaches have been used to parallelize these OOC DM algorithms by partitioning data among the machines of the cluster. This makes it possible to scale-up the system since storage and computation power are increased, and to exploit a larger I/O bandwidth since multiple disks interfaced with multiple buses are employed. We also developed dynamic techniques, such as load balancing and adaptive modification of the degree of parallelism, to take into account system heterogeneity and improve system utilization.



The result of an experiment implementing an R&C problem. Total execution time and I/O+idle time are plotted, varying the computational grain.

Our next goal will be to extend the DM set of algorithms tested against the general results found so far. We are now studying a classification algorithm for building decision trees and algorithms for data mining on the Web. We intend to build a set of methodologies for DM I/O intensive algorithms, characterized by application

independence and run time adaptability. The methodologies should be implemented as a general tool in an Application Programming Interface.

Links:
http://miles.cnuce.cnr.it/~palmeri/datam

Please contact:
Paolo Palmerini - CNUCE-CNR
Tel: +39 050 315 2967
E-mail: paolo.palmerini@cnuce.cnr.it