Input/Output Access Pattern Classification

Abstract

Input/output can be a critical performance bottleneck for many important scientific applications within Shell. To study application access pattern information, we propose investigating an automatic input/output access pattern classification technique. Ideally, such an automatic classification can optimise time required to process input/output requests and thus increase application performance. Our proposal is to use hidden Markov models for classifying input/output access patterns, using traces from previous application executions. This method in our opinion will increase resource utilization and will provide more precise control over caching and pre-fetching, leading to a faster more robust application performance.

1 Access Pattern Classification

A file access pattern is a description of past and future accesses (e.g., read-only, sequential, with a request size of 8 KB). Most current operating systems are optimised for a small number of common access patterns and do not modify caching or pre-fetching policies when non-standard pattern occurs. For example, most UNIX operating systems are optimised for sequential access patterns and perform poorly for random access.

Thus automatic recognition of access patterns and further utilization should improve application performance, specifically when it comes to large and complex data being worked and even more so in combination with latency (network or otherwise) to that data.

File access pattern description does not require being a perfect sequence of future access requests. It simply needs to contain enough information to select suitable caching or pre-fetching policies.

A model that allows calculating and expressing probability of accessing a given portion of a file in the future, having observed some access sequence, can supply sufficient access pattern information for choosing best suited policy. The function that we propose using is the probability distribution function of future accesses, based on some observed access sequence.

2 Presence of Access Patterns

A feasibility test was performed to determine whether there are actually detectable patterns in at least one of Shell's scientific applications. The tests were done with EP's "123di" software, which provides functionality for visualising and annotating seismic data. The data access pattern of this application is determined by the intersection of one or more 2-dimensional planes that are being looked at by the user, with 3-dimensional seismic data that is serialized in a disk file. By looking at the IO requests during a typical use session, we determined that there is indeed structure in the access pattern. The pattern in case could be described as a "strided read", where small extents are read, separated by an offset that is a few orders of magnitude larger. The figure below shows a histogram of the offsets between subsequent reads. The sharp peaks indicate a strong structure.

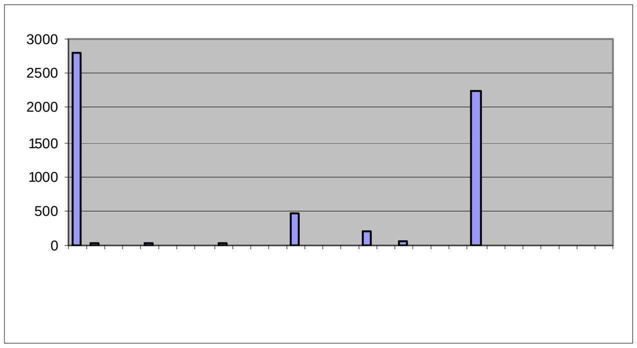

Figure 1: a histogram of offsets between successive reads for the "123di" application.

3 Hidden Markov Models

A discrete-time Markov process is a system that at any time is in one of N distinct states. At discrete time, the system changes states, or makes transitions, according to a set of probabilities associated with each state. Each state corresponds to a single observable event, or an observation symbol.

In discrete Markov process, the observation symbol uniquely determines the next state. A hidden Markov process is a generalization of a discrete Markov process where a given state may have several exiting transitions, all corresponding to the same observation symbol.

Hidden Markov models can learn a hidden behaviour of the application that generates input/output requests from observing the access request stream. Efficient algorithms do exist to train such models on observed sequences and dynamically calculate probability of future sequences.

In hidden Markov models a pattern change is possible only when the probability of accessing the next anticipated segment of data within a file using the current pattern falls bellow some specified threshold. Thus, hidden Markov model can be used to monitor the input/output stream, automatically recognizing access pattern changes. This dynamic classification provides an important ability to refine caching and pre-fetching policies as the access pattern changes.

4 Proposal

We propose to develop a proof-of-concept implementation of the Hidden Markov Model for data pre-fetching on the Linux operating system. The choice for Linux is based on the fact that a large part of Shell's scientific application portfolio is supported on this platform. In addition, the Linux kernel provides a debugging interface that makes it possible to experiment with different implementations, without the need to change the operating system itself or the application.

5 Conclusion

As mentioned earlier ability to predict access patterns is a way of improving performance of critical applications, such as the once used for working 3D seismic data. Decreasing latency and increasing throughput are the key advantages of the ability to predict input/output request sequences.