Abstract

An understanding of Input/Output data access patterns of applications is useful in several situations. First, gaining an insight into what applications are doing with their data at a semantic level helps in designing efficient storage systems. Second, it helps to create benchmarks that mimic realistic application behavior closely. Third, it enables autonomic systems as the information obtained can be used to adapt the system in a closed loop.

All these use cases require the ability to extract the application-level semantics of I/O operations. Methods such as modifying application code to associate I/O operations with semantic tags are intrusive. It is well known that network file system traces are an important source of information that can be obtained non-intrusively and analyzed either online or offline. These traces are a sequence of primitive file system operations and their parameters. Simple counting, statistical analysis or deterministic search techniques are inadequate for discovering application-level semantics in the general case, because of the inherent variation and noise in realistic traces.

In this paper, we describe a trace analysis methodology based on Profile Hidden Markov Models. We show that the methodology has powerful discriminatory capabilities that enables it to recognize applications based on the patterns in the traces, and to mark out regions in a long trace that encapsulate sets of primitive operations that represent higher-level application actions. It is robust enough that it can work around discrepancies between training and target traces such as in length and interleaving with other operations. We demonstrate the feasibility of recognizing patterns based on a small sampling of the trace, enabling faster trace analysis. Preliminary experiments show that the method is capable of learning accurate profile models on live traces in an online setting. We present a detailed evaluation of this methodology in a UNIX environment using NFS traces of selected commonly used applications such as compilations as well as on industrial strength benchmarks such as TPC-C and Postmark, and discuss its capabilities and limitations in the context of the use cases mentioned above.