A Metadata Workload Generator for Data-Intensive File Systems

Cristina L. Abad (student presenter, UIUC), Kihwal Lee (Yahoo!), Nathan Roberts (Yahoo!), Yi Lu (UIUC), Roy H. Campbell (UIUC)

Large-scale data-intensive computing [2, 3] has posed numerous challenges to the underlying distributed file system, due to the unprecedented amount of data, the large number of users, the intense competition on cost and service quality, and the emergence of new applications. As a result, there has been an increasing amount of research on scalable metadata management [4, 6], high availability [6], efficient scheduling [7], and dynamic data replication and placement [1].

The performance of the proposed solutions depends on the characteristics of the metadata workload, which describes file operations with respect to the underlying namespace, and captures correlation across time and file system structures. One example where realistic metadata workloads are highly desirable is the design of peta and exascale data-intensive file systems [5, 6]. The large scale of the system requires partitioning of the namespace, whose performance depends on the nature of the workload. Existing micro-benchmarks such as `mdtest` assume uniform data access patterns across all files, while realistic workloads, such as those we analyzed, exhibit highly skewed access patterns both in terms of file sizes and time since creation. The discrepancy in assumptions can cause ineffective namespace partitioning.

Currently there exist no publicly available traces from large-scale data-intensive systems. Designers and evaluators often rely on proprietary traces and disclose only the characteristics of workloads that are interesting to that particular work. Comparisons of solutions to the same problem are difficult as the characteristics of one workload are difficult to verify in others.

Within the Internet-services data-intensive community a few benchmark tools have emerged, but they have targeted other layers of the infrastructure (e.g., Gridmix for map-reduce and YCSB for key-value stores).

We present a tool that generates representative workloads based on trace studies on two large production clusters at Yahoo!, and user-configurable workloads based on proprietary traces and/or user-specified parameters. The synthetic workloads generated are guaranteed to be statistically realistic with respect to the original traces, and conform to parameters set by users.

The Dataset. We analyzed the access patterns of one month (May 2011) of two large Yahoo! clusters. We mined the HDFS [5] namenode logs which contain every metadata operation (open, create, mkdir, etc.) on the cluster. The PROD cluster has roughly 4000 nodes, and is the largest production cluster at Yahoo!. R&D is a 1900-node research and development cluster used for ad hoc queries, business intelligence, and large-scale testing of products before they go to production.

Design

Our tool consists of the statistical analysis engine (SAE) and the workload generation engine (WGE). The SAE analyzes proprietary traces and extracts a set of statistical parameters. The WGE takes the stored parameters and generates synthetic workloads that maintain the statistical parameters of the original traces.

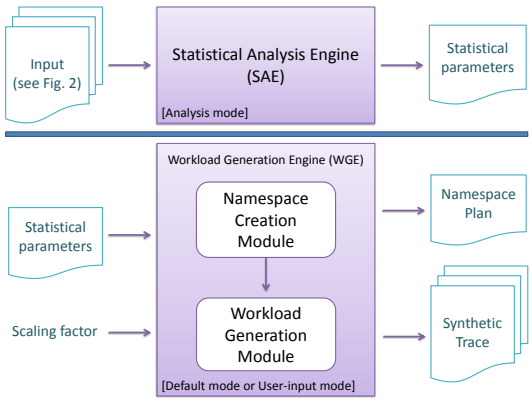

Figure 1: Block diagram of the components of our tool.

The SAE takes processed logs in a specified format as input, and outputs extracted parameters. We used the SAE to extract the parameters from two large production traces and stored them with our tool (default).

The WGE has two modes: the *Default* mode and the *User-input* mode. The *Default* mode uses parameters extracted from the production traces and stored with the framework to generate representative workloads statistically similar to the production traces. The *User-input* mode takes the default parameters, or parameters extracted from user-input traces, both of which are configurable through a simple command-line interface. Workloads conforming to these parameters are statistically generated. The WGE is further divided into two modules, corresponding to two parts of the workload generation. The namespace creation module generates the underlying file system structures with the given distributions and the workload generation module generates the synthetic trace with the given access patterns.

There are three main sets of parameters that characterize the workload (Table 1): the namespace structure, the access pattern, and the workload characteristics, including the workload-induced namespace.

The namespace structure is extracted from a snapshot

Table 1: Statistical parameters.

Namespace characterization
Number of directories *and* number of files
Distribution of files at each depth in namespace hierarchy
Distribution of directories at each depth in namespace hierarchy
Distribution of number of files per directory
Distribution of number of subdirectories per directory
Distribution of file sizes (in MB)
Access pattern characterization
Distribution of file age at time of access
Distribution of file age at deletion (i.e., file life span)
Workload characterization
Percentiles of operation type in trace
Interarrival rate distribution
Percentiles of operations observed at each depth in namespace
Distribution of files per depth in namespace, as observed in trace
Distribution of dirs. per depth in namespace, as observed in trace
Distribution of number of files per directory, as observed in trace
Distribution of number of subdirs. per dir., as observed in trace

of the namespace image and describes the shape of the namespace hierarchy tree. The number of directories and number of files describe the size of the namespace, which are not preserved when scaling the namespace onto a different cluster. The shape of the hierarchy tree are described by the following distributions: the number of files at each depth in the namespace hierarchy, the number of directories at each depth, the number of files per directory and the number of subdirectories per directory. These shapes are preserved when generating synthetic namespace on a cluster with a different size. Note that these distributions are not independent, hence fitting a workload to all of the distributions requires the solution of a bin-packing problem. In addition, the file size distribution is also extracted, which can be important to problems involving data block replication and placement.

The access patterns describe the relationship between the operations and the age of the files, which indirectly describe temporal correlation. This is important, for instance, to namespace partitioning or metadata caching. The distribution of the file age at the time of access and that at the time of deletion are extracted, and are reproduced in synthetic workloads.

The workload-induced namespace describes the hierarchy of files that actually occur in the trace. The shape of the hierarchy tree induced by the workload can be significantly different from that obtained from a snapshot when there are a large number of rarely accessed files. The hierarchy of the actual accessed files is important to metadata caching, for instance. In addition, the percentile of operation types and interarrival rates are also extracted and reproduced in the synthetic workload.

Each of the distributions listed in the parameter table takes value as either a known distribution or a percentile table. The SAE attempts to fit a known distribution to the measured values. If the values pass the goodness-of-fit test, the known distribution becomes the value of the parameter; otherwise, an empirical distribution is built using the cumulative distribution function (CDF) of the percentiles observed in the input data.

Case Study: Metadata cache for HDFS

To evaluate the usefulness of the traces generated by our tool, we tackle a real problem in the HDFS [5]: the need for a metadata cache for the namespace server. The expected effectiveness of a metadata cache could be evaluated with: (a) a real metadata workload trace, (b) a synthetic metadata workload trace, or (c) in the absence of traces, data sampled from an assumed distribution of popularity of data (e.g. Zipf). We found that the hit rate achieved by our metadata cache for varying cache sizes is comparable for the cases of real and synthetic traces generated with our tool. We also show that sampling from a known distribution (case (c) above) is a poor alternative due to the lack of temporal locality present in the sample.

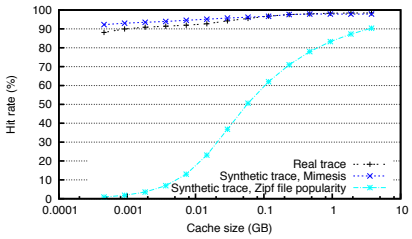

Figure 2: Hit rate for varying metadata sizes, using three 2-hour traces.

References

[1] ABAD, C., LU, Y., AND CAMPBELL, R. DARE: Adaptive data replication for efficient cluster scheduling. In *IEEE Conf. Cluster Comp.* (2011).

[2] DEAN, J., AND GHEMAWAT, S. MapReduce: Simplified data processing on large clusters. In *USENIX OSDI* (2004), pp. 137–150.

[3] ISARD, M., BUDIU, M., YU, Y., BIRRELL, A., AND FETTERLY, D. Dryad: Distributed data-parallel programs from sequential building blocks. In *Eur. Conf. Comput. Syst. (EuroSys)* (2007).

[4] NICOLAE, B., ANTONIU, G., BOUGÉ, L., MOISE, D., AND CARPEN-AMARIE, A. BlobSeer: Next Generation Data Management for Large Scale Infrastructures. *J. Par. and Distrib. Comp. 71*, 2 (2011).

[5] SHVACHKO, K. V. HDFS scalability: The limits to growth. *;login: USENIX Magazine 35*, 2 (2010).

[6] WEIL, S., BRANDT, S., MILLER, E., LONG, D., AND MALTZAHN, C. Ceph: A scalable, high-performance distributed file system. In *OSDI 2006*.

[7] ZAHARIA, ET AL. Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling. In *EuroSys* 2010.