Specifying Trace Data Collection, Validation and Access

Objective results are hard to achieve when data is scarce, relevant benchmarks are few and validation of the data/traces that do exist have not been typically done. This is the situation that storage researchers today find themselves in. For example, a widely quoted decade old trace data that purports to be over multiple months can barely even represent a single day as the time field has been truncated erroneously to 32 bits.

We discovered this in the course of our research on how to optimize/minimize migration of local storage state such as synchronization of data across multiple devices or VM migration that includes storage state as well. We attempted to use three different well regarded, publicly available traces and not one of them served the purpose. In fact, it needed significant effort to establish them as unusable due to the kind of problems mentioned above.

Our attempts to generate traces were centred on the use of SystemTap but we found it difficult to deploy it beyond 7-10 machines due to the heterogeneity of kernels and associated instability. Since these traces are large and difficult to generate in the first place, we believe some standardization effort is needed in the community to make generation, validation and documentation of the traces or benchmark data easier.

We see the need for a tracing or data monitoring tool that is independent of the kernel (DTrace is an excellent example) complemented by ease of customization and high level specification that can be used to check if the trace data indeed follows the spec. The spec can be fairly straightforward like a timefield being able to represent multiple years (32 bits vs 64 bits for 100 ns resolution of events) or more systemic ones like whether privacy guarantees are honored. Also, it must be possible to achieve scalability while keeping the techniques flexible enough to configure the setup so as to capture only relevant data items.

Since most operating systems have applications for backing up data, applying privacy preserving functions and making them available for use can make crowd-sourcing a feasible option to collect data.

In summary, we need tools to collect data scalably and be able to validate collected data easily. The data collection must be modular and able to apply privacy-preserving functions. The Linux kernel (along with userspace applications) is an ideal platform as it forms the basis for several operating systems.

---

1 Two of these were used in papers that appeared in FAST and another in SIGMETRICS.