Representative Probes

We take MySQL as an example to illustrate several representative probes used by DBdoctor for fine-grained metric collection.

(1) *How to collect I/O related metrics for each SQL?* To collect per-SQL I/O consumption, DBdoctor attaches eBPF probes to the kernel-level functions *vfs_write()* and *vfs_read()*. For each invocation, the probe extracts the process ID, thread ID, file identifier, I/O byte size, and timestamp. To associate I/O operations with specific SQL statements, DBdoctor first instruments the SQL entry function (e.g., *dispatch_command()*) to obtain the thread ID, SQL identifier, and the start timestamp of the SQL execution. In addition, probes attached to I/O-related functions also record the thread ID, file identifier, and timestamp. DBdoctor then correlates these messages based on thread ID, file identifier, and timestamps, thereby attributing the observed I/O consumption to individual SQL statements.

(2) *How to collect network usage for each SQL?* To collect network-level resource usage, DBdoctor attaches probes to the kernel functions *tcp_sendmsg()* and *tcp_cleanup_rbuf()*, which correspond to network send and receive operations, respectively. For each invocation, the probes collect the process ID, thread ID, timestamp, and the size of transmitted or received data, which is obtained from the function return value. Similarly, DBdoctor attaches probes to the SQL entry function and data sending/receiving functions to obtain the thread ID, SQL identifier, and timestamps. These messages are then associated with SQL executions using thread ID and timestamp, enabling per-SQL network usage attribution.

(3) *Other Metrics.* DBdoctor collects CPU consumption of each SQL by attaching probes to computation-related functions, such as scan, aggregation, and sort operators in the query execution engine. Memory consumption is collected by monitoring functions responsible for buffer pool usage, and per-connection memory allocation functions (e.g., buffer management routines), etc. Lock-related metrics are collected by attaching probes to lock acquisition, waiting, and release functions.