DEBUG LOGS

The main debugging tool for Ceph is the dout and derr logging functions. Collectively, these are referred to as “dout logging.”

Dout has several log faculties, which can be set at various log levels using the configuration management system. So it is possible to enable debugging just for the messenger, by setting debug_ms to 10, for example.

Dout is implemented mainly in common/DoutStreambuf.cc

The dout macro avoids even generating log messages which are not going to be used, by enclosing them in an “if” statement. What this means is that if you have the debug level set at 0, and you run this code:

```
dout(20) << "myfoo() = " << myfoo() << dendl;
```

myfoo() will not be called here.

Unfortunately, the performance of debug logging is relatively low. This is because there is a single, process-wide mutex which every debug output statement takes, and every debug output statement leads to a write() system call or a call to syslog(). There is also a computational overhead to using C++ streams to consider. So you will need to be parsimonius in your logging to get the best performance.

Sometimes, enabling logging can hide race conditions and other bugs by changing the timing of events. Keep this in mind when debugging.

PERFORMANCE COUNTERS

Ceph daemons use performance counters to track key statistics like number of inodes pinned. Performance counters are essentially sets of integers and floats which can be set, incremented, and read using the PerfCounters api.

A PerfCounters object is usually associated with a single subsystem. It contains multiple counters. This object is thread-safe because it is protected by an internal mutex. You can create multiple PerfCounters objects.

Currently, three types of performance counters are supported: u64 counters, float counters, and long-run floating-point average counters. These are created by PerfCountersBuilder::add_u64, PerfCountersBuilder::add_fl, and PerfCountersBuilder::add_fl_avg, respectively. u64 and float counters simply provide a single value which can be updated, incremented, and read atomically. floating-pointer average counters provide two values: the current total, and the number of times the total has been changed. This is intended to provide a long-run average value.

Performance counter information can be read in JSON format from the administrative socket (admin_sock). This is implemented as a UNIX domain socket. The Ceph peformance counter plugin for collectd shows an example of how to access this information. Another example can be found in the unit tests for the administrative sockets.