NAME

statistics – Shore Storage Manager performance information

SYNOPSIS

```
class sm_stats_info_t;

static rc_t
ss_m::gather_stats(sm_stats_info_t& stats, bool reset=false);
```

DESCRIPTION

The Shore Storage Manager keeps myriad statistics. The statistics are available to higher layers of software through the method **gather_stats.** Each statistic kept has a name and a manifest constant of that name for use with the generic statistics-gathering programming interface described in **statistics(fc).** The names and descriptions of the statistics are listed in sections below.

All the statistics are unsigned long integers except where noted below. Below each statistic is identified by the manifest constant for the statistic and also by its member name in the structure *sm_stats_info_t.*

After gathering the storage manager statistics into a *sm_stats_info_t*, the *sm_stats_info_t* can be used with the *w_statistics_t* package for display:

```
sm_stats_info_t s;

rc =  gather_stats(s);
if(!rc) {
      w_statistics_t stats << s;
      cout << stats;
}
```

Set the argument *reset* of **gather_stats** to **true** if you want the statistics to be cleared (set to zero) after they are copied.

BUFFER POOL STATISTICS

SM_rec_pin_cnt
: *sm_stats_info_t::rec_pin_cnt.* Records pinned in the buffer pool. (This counts the pinning events, not the distinct records pinned.)

SM_rec_unpin_cnt
: *sm_stats_info_t::rec_unpin_cnt.* Records unpinned.

SM_page_fix_cnt
: *sm_stats_info_t::page_fix_cnt.* Times pages were fixed in the buffer pool.

SM_page_refix_cnt
: *sm_stats_info_t::page_refix_cnt.* Times pages were refixed in the buffer pool; refixing is cheaper than fixing, because it avoids a hash-table look-up.

SM_page_unfix_cnt
: *sm_stats_info_t::page_unfix_cnt.* Times pages were unfixed.

The following group of statistics describes the operation of the buffer manager, which groups consecutive pages for writing to disk whenever possible.

SM_bf_one_page_write
: *sm_stats_info_t::bf_one_page_write.* The number of times a single page was flushed from the buffer pool.

SM_bf_two_page_write
sm_stats_info_t::bf_two_page_write. The number of times two consecutive pages were flushed from the buffer pool in one I/O request.

SM_bf_three_page_write
sm_stats_info_t::bf_three_page_write.

SM_bf_four_page_write
sm_stats_info_t::bf_four_page_write.

SM_bf_five_page_write
sm_stats_info_t::bf_five_page_write.

SM_bf_six_page_write
sm_stats_info_t::bf_six_page_write.

SM_bf_seven_page_write
sm_stats_info_t::bf_seven_page_write.

SM_bf_eight_page_write
sm_stats_info_t::bf_eight_page_write.

SM_cleaner_sweeps
sm_stats_info_t::bf_cleaner_sweeps. The number of times the buffer-cleaner thread swept the buffer pool (a clock algorithm is used).

SM_bf_log_flush_all
sm_stats_info_t::bf_log_flush_all. The number of times the entire log was flushed by the buffer manager.

SM_bf_log_flush_lsn
sm_stats_info_t::bf_log_flush_lsn. The number of times the log was flushed up to a specific log sequence number by the buffer manager.

SM_bf_write_out
sm_stats_info_t::bf_write_out. The number of calls to the buffer manager method that writes pages to disk.

SM_bf_replace_out
sm_stats_info_t::bf_replace_out. The number of times a page was written in order to free the frame (during replacement) for a different page.

SM_replaced_dirty
sm_stats_info_t::bf_replaced_dirty. The number of times the frame replaced contained a dirty page.

SM_replaced_clean
sm_stats_info_t::bf_replaced_clean. The number of times the frame replaced contained a clean page.

SM_await_clean
sm_stats_info_t::bf_await_clean. The number of times a page fix request awaited a frame to become clean by the cleaner thread, rather than forcing out a dirty page.

SM_cleaner_signalled
sm_stats_info_t::bf_cleaner_signalled. The number of times the cleaner was restarted by a signal from another thread. The following three counters distinguish the reasons for the signals.

SM_kick_replacement
sm_stats_info_t::bf_kick_replacement. A dirty page was replaced.

SM_kick_full
sm_stats_info_t::bf_kick_full. The buffer pool was full of dirty pages, noticed sometime other than on page replacement.

SM_kick_threshhold
sm_stats_info_t::bf_kick_threshhold. The buffer pool's dirty-page threshhold was met.

SM_sweep_page_hot
sm_stats_info_t::bf_sweep_page_hot. The buffer cleaner swept over a hot page, and left it in place.

B-TREE STATISTICS

SM_bt_find_cnt
sm_stats_info_t::bt_find_cnt. B-tree lookups (calls to ss_m::find_assoc());

SM_bt_insert_cnt
sm_stats_info_t::bt_insert_cnt. B-tree inserts (calls to ss_m::create_assoc());

SM_bt_remove_cnt
sm_stats_info_t::bt_remove_cnt. B-tree removes (calls to ss_m::destroy_assoc());

SM_bt_scan_cnt
sm_stats_info_t::bt_scan_cnt. Number of B-tree scans started.

SM_bt_splits
sm_stats_info_t::bt_splits. B-tree pages split (interior and leaf).

SM_bt_cuts
sm_stats_info_t::bt_cuts. B-tree pages removed (interior and leaf).

SM_bt_grows
sm_stats_info_t::bt_grows. Times B-tree grew a level.

SM_bt_shrinks
sm_stats_info_t::bt_shrinks. Times B-tree shrunk a level.

SM_bt_links
sm_stats_info_t::bt_links. Times B-tree sibling links were followed (while a structure modification operation was not yet propagated.)

SM_bt_clr_smo_traverse
sm_stats_info_t::bt_clr_smo_traverse. Times SMO (structure-modification-operation-in-progress) bits cleared on traverse.

SM_bt_posc
sm_stats_info_t::bt_posc. Times awaited a POSC (point of structural consistency).

SM_bt_traverse_cnt
sm_stats_info_t::bt_traverse_cnt. Times b-trees were traversed from the root.

SM_bt_upgrade_fail_retry
sm_stats_info_t::bt_upgrade_fail_retry. Failure to upgrade a latch without waiting caused a re-try.

LOGICAL-ID STATISTICS

These will be zero if you do not use logical-IDs. Logical IDs are stored in a B-tree index, and recently used IDs are cached in a transient cache.

SM_lid_lookups
sm_stats_info_t::lid_lookups. Times the logical-ID index was searched for an ID.

SM_lid_remote_lookups
sm_stats_info_t::lid_remote_lookups. Times the index was searched for a second time for indirect references.

SM_lid_inserts
sm_stats_info_t::lid_inserts. Logical-IDs added to the index.

SM_lid_removes
sm_stats_info_t::lid_removes. Logical-IDs removed from the index.

SM_lid_cache_hits
sm_stats_info_t::lid_cache_hits. Times a logical-ID look-up request was satisfied by the cache of recent requests, avoiding a B-tree look-up.

PAGE & EXTENT OPERATIONS

SM_page_alloc_cnt
sm_stats_info_t::page_alloc_cnt. Pages allocated from free pages in allocated extents.

SM_page_dealloc_cnt
sm_stats_info_t::page_dealloc_cnt. Pages deallocated. These pages are free for re-allocation in the same store.

SM_ext_lookup_hits
sm_stats_info_t::ext_lookup_hits. Cache hits during extent-lookups.

SM_ext_lookup_hits
sm_stats_info_t::ext_lookup_hits. Cache misses during extent-lookups.

TRANSACTION STATISTICS

SM_begin_xct_cnt
sm_stats_info_t::begin_xct_cnt. The number of transactions that were begun. This includes the number that resulted from chaining transactions.

SM_commit_xct_cnt
sm_stats_info_t::commit_xct_cnt. The number of transactions that were committed. This includes the number that resulted from chaining transactions.

SM_abort_xct_cnt
sm_stats_info_t::abort_xct_cnt. The number of transactions that were aborted.

SM_rollback_savept_cnt
sm_stats_info_t::rollback_savept_cnt. The number of requests to roll back to a save point, without rolling back the entire transaction.

SM_mpl_attach_cnt
sm_stats_info_t::mpl_attach_cnt. Times a thread attached to a transaction to which at least one other thread was already attached. This is for value-added servers that run transactions with multiple threads in parallel. (This is not a supported feature, as the circumstances in which this can be done are few.)

SM_anchors
sm_stats_info_t::anchors. Times a transaction grabbed an anchor in the log (for the purpose of compensating around a top-level action).

SM_compensate_in_log
sm_stats_info_t::compensate_in_log. Times a transaction wrote a compensation by snooping into the log.

SM_compensate_in_xct
sm_stats_info_t::compensate_in_xct. Times a transaction wrote a compensation in its own log buffer.

SM_compensate_records
sm_stats_info_t::compensate_records. Number of compensations-only log records written.

PARALLELISM STATISTICS

SM_await_io_monitor
sm_stats_info_t::await_io_monitor. Times blocked on mutex for serializing access to the I/O monitor.

LOCK STATISTICS

The following statistics measure activity in the lock table. Many of the statistics are of interest only the the developers of Shore, and are likely to be removed in future releases of Shore.

SM_unlock_request_cnt
sm_stats_info_t::unlock_request_cnt. High-level unlock requests.

SM_lock_request_cnt
sm_stats_info_t::lock_request_cnt. High-level lock requests (could have been satisfied with the lock cache or with the lock table.)

SM_lock_cache_hit_cnt
sm_stats_info_t::lock_cache_hit_cnt. The number of cache hits (avoiding request to acquire locks through the lock table).

SM_lock_acquire_cnt
sm_stats_info_t::lock_acquire_cnt. The number of times a request was made to acquire a lock through the lock table (as opposed to the lock cache).

SM_lock_head_t_cnt
sm_stats_info_t::lock_head_t_cnt. The number of *lock_head_t* structures put in the table.

SM_lock_request_t_cnt
sm_stats_info_t::lock_request_t_cnt. The number of *lock_request_t* structures chained from a *lock_head_t.*

SM_lock_query_cnt
sm_stats_info_t::lock_query_cnt. The number of times the lock table was queried about a given lock.

The following statistics distinguish the locks acquired by lock-id:

SM_lk_vol_acq
sm_stats_info_t::lk_vol_acq. Volume locks.

SM_lk_store_acq
sm_stats_info_t::lk_store_acq. Store locks. Stores are below volumes in the lock hierarchy.

SM_lk_page_acq
sm_stats_info_t::lk_page_acq. Page locks. Pages are below stores in the lock hierarchy.

SM_lk_kvl_acq
sm_stats_info_t::lk_kvl_acq. Key-value locks. Key-value pairs are below pages in the lock hierarchy.

SM_lk_rec_acq
sm_stats_info_t::lk_rec_acq. Record locks. Records are below pages in the lock hierarchy.

SM_lk_ext_acq
sm_stats_info_t::lk_ext_acq. Extent locks. Extents are not in the lock hierarchy.

SM_lock_deadlock_cnt
sm_stats_info_t::lock_deadlock_cnt. Deadlocks detected by local deadlock detector.

SM_lock_esc_to_page
sm_stats_info_t::lock_esc_to_page. Lock escalations from record to page.

SM_lock_esc_to_store
sm_stats_info_t::lock_esc_to_store. Lock escalations from page to store.

SM_lock_esc_to_volume
sm_stats_info_t::lock_esc_to_volume. Lock escalations from store to volume.

The following statistics measure collisions from the lock table hash function. They are generally not of interest to the user, but they might be of interest to someone who is building a value-added server and might consider using a different hash function. Only buckets of non-zero length are counted when these statistics are computed. They are all zero when there is no transaction active, since the lock table is empty at that time. These statistics are computed when **gather_stats** is called; it traverses the entire lock table, so it should not be used habitually when transactions are active.

SM_lock_conversion_cnt
sm_stats_info_t::lock_conversion_cnt. Lock requests requiring a conversion of the lock mode.

SM_lock_extraneous_req_cnt
sm_stats_info_t::lock_extraneous_req_cnt. Requests already granted.

sm_locktablesize.

SM_lock_max_bucket_len
sm_stats_info_t::lock_max_bucket_len. The largest bucket in use.

SM_lock_min_bucket_len
sm_stats_info_t::lock_min_bucket_len. The smallest bucket in use.

SM_lock_mode_bucket_len
sm_stats_info_t::lock_mode_bucket_len. The mode of the lengths of the buckets used.

SM_lock_mean_bucket_len
float sm_stats_info_t::lock_mean_bucket_len. The mean bucket length (of the buckets used).

SM_lock_var_bucket_len
float sm_stats_info_t::lock_var_bucket_len. The variance of the bucket length (of the buckets used).

SM_lock_std_bucket_len
float sm_stats_info_t::lock_std_bucket_len. The Standard deviation of the bucket length (of the buckets used).

OPERATIONS ON LOCAL DATA VOLUMES

SM_vol_reads
sm_stats_info_t::vol_reads. Lowest-level read requests made to the ‘diskrw’ process, which effects non-blocking disk I/O for data volumes.

SM_vol_writes
sm_stats_info_t::vol_writes. Lowest-level write requests made to the ‘diskrw’ process, which effects non-blocking disk I/O for data volumes.

SM_vol_blks_written
sm_stats_info_t::vol_blks_written. Data volume pages written to disk.

OPERATIONS ON THE LOG

SM_log_records_generated
sm_stats_info_t::log_records_generated. The number of log records written.

SM_log_bytes_generated
sm_stats_info_t::log_bytes_generated. The number of bytes written to the log.

SM_log_sync_cnt
sm_stats_info_t::log_sync_cnt. The number of times the log was flushed to disk.

SM_log_dup_sync_cnt
sm_stats_info_t::log_dup_sync_cnt. The number of times the log was flushed superfluously (for debugging).

SM_log_fsync_cnt
sm_stats_info_t::log_fsync_cnt. The number of times the fsync(2) system call was used to flush a log that is a Unix file.

SM_log_sync_nrec_max
sm_stats_info_t::log_sync_nrec_max. Maximum number log records buffered between flushes.

SM_log_sync_nbytes_max
sm_stats_info_t::log_sync_nbytes_max. Maximum number bytes buffered between log flushes.

SM_log_chkpt_cnt
sm_stats_info_t::log_chkpt_cnt. Number of checkpoints taken.

MISCELLANEOUS

SM_idle_yield_return
sm_stats_info_t::idle_yield_return. Times the idle thread returned from yield(). (For debugging.)

SM_idle_wait_return
Times the idle thread returned from wait(). (For debugging.) *sm_stats_info_t::idle_wait_return.*

VERSION

This manual page applies to Version 2.0 of the Shore Storage Manager.

SPONSORSHIP

The Shore project is sponsored by the Advanced Research Project Agency, ARPA order number 018 (formerly 8230), monitored by the U.S. Army Research Laboratory under contract DAAB07-91-C-Q518. Further funding for this work was provided by DARPA through Rome Research Laboratory Contract No. F30602-97-2-0247.

SEE ALSO

options(svas), statistics(svas), and **statistics(fc)**