NAME

dtrace_lockstat - a DTrace provider for tracing CPU scheduling events

SYNOPSIS

lockstat:::adaptive-acquire(*struct mtx **);

lockstat:::adaptive-release(*struct mtx **);

lockstat:::adaptive-spin(*struct mtx *, uint64_t*);

lockstat:::adaptive-block(*struct mtx *, uint64_t*);

lockstat:::spin-acquire(*struct mtx **);

lockstat:::spin-release(*struct mtx **);

lockstat:::spin-spin(*struct mtx *, uint64_t*);

lockstat:::rw-acquire(*struct rwlock *, int*);

lockstat:::rw-release(*struct rwlock *, int*);

lockstat:::rw-block(*struct rwlock *, uint64_t, int, int, int*);

lockstat:::rw-spin(*struct rwlock *, uint64_t*);

lockstat:::rw-upgrade(*struct rwlock **);

lockstat:::rw-downgrade(*struct rwlock **);

lockstat:::sx-acquire(*struct sx *, int*);

lockstat:::sx-release(*struct sx *, int*);

lockstat:::sx-block(*struct sx *, uint64_t, int, int, int*);

lockstat:::sx-spin(*struct sx *, uint64_t*);

lockstat:::sx-upgrade(*struct sx **);

lockstat:::sx-downgrade(*struct sx **);

lockstat:::lockmgr-acquire(*struct lock **, *int*);

lockstat:::lockmgr-release(*struct lock **, *int*);

lockstat:::lockmgr-disown(*struct lock **, *int*);

lockstat:::lockmgr-block(*struct lock **, *uint64_t*, *int*, *int*, *int*);

lockstat:::lockmgr-upgrade(*struct lock **);

lockstat:::lockmgr-downgrade(*struct lock **);

lockstat:::thread-spin(*struct mtx **, *uint64*);

DESCRIPTION

The DTrace **lockstat** provider allows the tracing of events related to locking on FreeBSD.

The **dtrace_lockstat** provider contains DTrace probes for inspecting kernel lock state transitions. Probes exist for the lockmgr(9), mutex(9), rwlock(9), and sx(9) lock types. The lockstat(1) utility can be used to collect and display data collected from the **dtrace_lockstat** provider. Each type of lock has **acquire**() and **release**() probes which expose the lock structure being operated upon, as well as probes which fire when a thread contends with other threads for ownership of a lock.

The **lockstat:::adaptive-acquire**() and **lockstat:::adaptive-release**() probes fire when an MTX_DEF mutex(9) is acquired and released, respectively. The only argument is a pointer to the lock structure which describes the lock being acquired or released.

The **lockstat:::adaptive-spin**() probe fires when a thread spins while waiting for a MTX_DEF mutex(9) to be released by another thread. The first argument is a pointer to the lock structure that describes the lock and the second argument is the amount of time, in nanoseconds, that the mutex spent spinning. The **lockstat:::adaptive-block**() probe fires when a thread takes itself off the CPU while trying to acquire an MTX_DEF mutex(9) that is owned by another thread. The first argument is a pointer to the lock structure that describes the lock and the second argument is the length of time, in nanoseconds, that the waiting thread was blocked. The **lockstat:::adaptive-block**() and **lockstat:::adaptive-spin**() probes fire only after the lock has been successfully acquired, and in particular, after the **lockstat:::adaptive-acquire**() probe fires.

The **lockstat:::spin-acquire**() and **lockstat:::spin-release**() probes fire when a MTX_SPIN mutex(9) is

acquired or released, respectively. The only argument is a pointer to the lock structure which describes the lock being acquired or released.

The **lockstat:::spin-spin**() probe fires when a thread spins while waiting for a MTX_SPIN mutex(9) to be released by another thread. The first argument is a pointer to the lock structure that describes the lock and the second argument is the length of the time spent spinning, in nanoseconds. The **lockstat:::spin-spin**() probe fires only after the lock has been successfully acquired, and in particular, after the **lockstat:::spin-acquire**() probe fires.

The **lockstat:::rw-acquire**() and **lockstat:::rw-release**() probes fire when a rwlock(9) is acquired or released, respectively. The first argument is a pointer to the structure which describes the lock being acquired. The second argument is 0 if the lock is being acquired or released as a writer, and 1 if it is being acquired or released as a reader. The **lockstat:::sx-acquire**() and **lockstat:::sx-release**(), and **lockstat:::lockmgr-acquire**() and **lockstat:::lockmgr-release**() probes fire upon the corresponding events for sx(9) and lockmgr(9) locks, respectively. The **lockstat:::lockmgr-disown**() probe fires when a lockmgr(9) exclusive lock is disowned. In this state, the lock remains exclusively held, but may be released by a different thread. The **lockstat:::lockmgr-release**() probe does not fire when releasing a disowned lock. The first argument is a pointer to the structure which describes the lock being disowned. The second argument is 0, for compatibility with **lockstat:::lockmgr-release**().

The **lockstat:::rw-block**(), **lockstat:::sx-block**(), and **lockstat:::lockmgr-block**() probes fire when a thread removes itself from the CPU while waiting to acquire a lock of the corresponding type. The **lockstat:::rw-spin**() and **lockstat:::sx-spin**() probes fire when a thread spins while waiting to acquire a lock of the corresponding type. All probes take the same set of arguments. The first argument is a pointer to the lock structure that describes the lock. The second argument is the length of time, in nanoseconds, that the waiting thread was off the CPU or spinning for the lock. The third argument is 0 if the thread is attempting to acquire the lock as a writer, and 1 if the thread is attempting to acquire the lock as a reader. The fourth argument is 0 if the thread is waiting for a reader to release the lock, and 1 if the thread is waiting for a writer to release the lock. The fifth argument is the number of readers that held the lock when the thread first attempted to acquire the lock. This argument will be 0 if the fourth argument is 1.

The **lockstat:::lockmgr-upgrade**(), **lockstat:::rw-upgrade**(), and **lockstat:::sx-upgrade**() probes fire when a thread successfully upgrades a held lockmgr(9), rwlock(9), or sx(9) shared/reader lock to an exclusive/writer lock. The only argument is a pointer to the structure which describes the lock being acquired. The **lockstat:::lockmgr-downgrade**(), **lockstat:::rw-downgrade**(), and **lockstat:::sx-downgrade**() probes fire when a thread downgrades a held lockmgr(9), rwlock(9), or sx(9) exclusive/writer lock to a shared/reader lock.

The **lockstat:::thread-spin**() probe fires when a thread spins on a thread lock, which is a specialized

MTX_SPIN mutex(9). The first argument is a pointer to the structure that describes the lock and the second argument is the length of time, in nanoseconds, that the thread was spinning.

SEE ALSO

dtrace(1), lockstat(1), locking(9), mutex(9), rwlock(9), SDT(9), sx(9)

HISTORY

The **dtrace_lockstat** provider first appeared in Solaris. The FreeBSD implementation of the **dtrace_lockstat** provider first appeared in FreeBSD 9.

AUTHORS

This manual page was written by George V. Neville-Neil <*gnn@FreeBSD.org*> and Mark Johnston <*markj@FreeBSD.org*>.

BUGS

Probes for rmlock(9) locks have not yet been added.