# S3F Barrier Types & Performance

# Barrier Types

- Original s3f barrier (barrier.cc)
- Mutex (broadcast) barrier (barrier_mutex.cc)
- Fast sense-reversing barrier (fast_barrier.cc)
  - Fast-userspace locking (futex) option
- Software combining tree (fast_tree_barrier.cc)
  - Arrival futex option
  - Release futex option
- Pthread barrier

# Original S3F Barrier

- Pros:
  - No locking at all
- Cons:
  - Freezes after 2^31 calls on systems with 32-bit longs (or 2^65 calls for 64-bit longs)
  - Potential invalid behavior on non-TSO (total-store-ordering) architectures (ie non-x86)
  - Extra unnecessary work by all but one thread
  - Poor cache performance due to memory contention

# Broadcast Barrier

- Uses pthread_cond_wait and pthread_cond_broadcast with a mutex lock
- Pros:
  - Sense-reversing (no call-limit)
  - Better cache behavior
- Cons:
  - VERY slow wakeup hinders s3f's performance

# 'Fast' Sense-Reversing Barrier

- Can use sched_yield() OR futex locking
  - Enable '#define USE_FUTEX' in fast_barrier.h
- Pros:
  - Sense-reversing (no call limit)
  - Optimized cache performance
  - Only locks to update sim window statistics
- Cons:
  - Can lead to high 'sys' (kernel-space) times (still better than original barrier) for both yield and futex

# Software Combining Tree

- Configurable with defaults:
  - ARRIVAL_CHILDREN = 4
  - RELEASE_CHILDREN = 2
- Can use sched_yield() or futex locking (defines in fast_tree_barrier.h):
  - For arriving children ('#define USE_ARRIVAL_FUTEX') – not recommended, generally increases 'sys' time overhead
  - For released children ('#define USE_RELEASE_FUTEX') – generally safe, but better if RELEASE_CHILDREN increased
- Pros:
  - Excellent for high numbers of threads
  - Potentially scalable to a distributed system
- Cons:
  - Extra lock overhead for low numbers of threads
  - Slightly slower release times
  - Futex locking does not appear to perform well

# Pthread Barrier

- Uses futex locking via assembly code
- Pros:
  - Ease of use – default pthread implementation
  - Excellent for large amounts of thread work
  - Very low (comparative) 'sys' time
- Cons:
  - Slightly slower wakeup than yield
  - Barrier not guaranteed in all pthread implementations
  - Manual window statistics updating required

# Using Alternative Barrier Types

- Enable exactly ONE in s3f.h:
  - '#define MUTEX_BARRIER' – mutex broadcast
  - '#define PTHREAD_BARRIER' – pthread
  - (0) '#define SCHED_YIELD_BARRIER' – all others
- Additional toggles:
  - (1) '#define FAST_SCHED_YIELD_BARRIER', requires (0)
  - (2) '#define TREE_BARRIER', requires (0) and (1)
  - (0) only → original barrier
  - (0) + (1) only → 'fast' sense-reversing barrier
  - (0) + (1) + (2) → software combining tree barrier
- Enable/disable futexes within *barrier* headers

# Performance Summary

- Pthread barrier adds the least overhead for almost all cases
- fast_barrier works well for very low thread work
- sched_yield() works best for fewer threads
- Futex locking works better for very small simulation synchronization windows (SSW)
  - WARNING: 'sys' time scales up exponentially as SSW increases
- fast_tree_barrier looks to work well for 32+ threads
  - Should be more extensible to a distributed system
- All performance looks to scale linearly with number of hosts
- Detailed performance data available in barrier_performance.xls (in barriers.tar.gz)