Reactive Synchronization
Lim and Agarwal

- Protocols
  - Test and Set
  - Queuing
- Waiting
  - Spinning*
  - Blocking
  - Competitive

Ski Rental Analogy

- Dynamically choose between two policies such that the performance will be within a constant factor of the optimal off-line choice
- Rent or buy skis?
  - Rent until you spend enough on rentals that you could have bought a pair, then buy
  - Spin until you spin long enough to “pay for” cost of context switch to block, then block

Test and Set Variations

- Dealing with contention of Test&Set spinlocks:
  - Don’t execute test&set so much
  - Spin without generating bus traffic
- Test&Set with Backoff
  - Insert delay between test&set operations (not too long)
  - Exponential seems good (k’*ci)
  - Not fair
- Test-and-Test&Set
  - Spin (test) on local cached copy until it gets invalidated, then issue test&set
  - Intuition: No point in trying to set the location until we know that it’s not set, which we can detect when it get invalidated...
  - Still contention after invalidate
  - Still not fair

Queue Lock Implementations

Mellor-Crummey and Scott (MCS locks)
Contestation

Baseline Performance

Reactive Lock Algorithm

• Only one of TTSL or MCSL will be free
• Mode variable is hint
• To acquire:
  – If mode appears to be TTS, spin with recheck of mode in the loop
  – If mode appears to be Queue, recheck if get a retry signal
• To release:
  – Change mode if appropriate
  – Queue to TTS, send retry signals
  – release whichever
• Change policy
  – TTS to Queue if # failed TS attempts > threshold
  – Queue to TTS if empty queue for some # of acquires
• Generalize as consensus object

Performance of Reactive Lock
Switching Overheads

Figure 9: The dynamic overheads vary inversely as the level of contention is increased, the slower lock or underlying cache change.