Practical Considerations for Non-Blocking Concurrent Objects

Non-blocking objects
Defn: concurrent object whose operations are not contained within mutually exclusive critical sections.
Not vulnerable to scheduling convoys, priority inversion, and deadlock.

Compare-and-Swap used to atomically update data.
- 3 args:
  - addr of shared data item
  - old value of shared data item
  - new value
- if current value == old value, data gets new value (SUCCESS)

Problems with Compare-and-Swap:
1. Few processors support.
2. Performance in presence of contention.

Problem 1:
- CAS must be implemented out of more basic primitives(test-and-set) offered by hardware.
- Problem is OS may preempt thread with lock effectively causing CAS to be blocking.
- OS support can eliminate the problem.
  - Roll-out:
    - At preemption, lock is forced to release thread.
    - On resumption, OS must clean up after preemption. (Make sure the right instruction is being executed.)
    - Drawbacks:
      - Assumes processor doesn't stop (fail) before release of lock.
      - Doesn't insure processor makes forward progress.

Problem 2:
- General trend of increasing relative cost of performing atomic operations with increasing processor speeds.
  - Need to decrease frequency of synchronization.
  - Because synchronization operations run relatively slower on faster processors, synchronized component will have larger effect on performance of faster processors.

Look at two possible implementations of lock
- Spinlocks
  - Problem is that ALL processors try to synchronize after the release of lock. This causes contention. A failed attempt to acquire lock also negatively affects all other processors, causing decreased throughput.
- Queue Locks

- Problem is that processes wait on the lock only to have their compare fail. They queue up and then are sure to fail since the last process obviously altered the value of the lock from the old value the current process had.

Solution:

- Compare-and-Compare-and-Swap
  - Processor tests old value for equality with data's current value while attempting to acquire the lock. If they are not the same, processor aborts. This way, processes aren't waiting on lock with old data values.
  - Throughput does not degrade as quickly as more processors are added.