Lock-free Programming Solutions

Programming with locks

- Briefly describe some of the disadvantages of programming with mutexes and locks
  - Race conditions caused by forgetting to lock, or using the wrong mutex
  - Lack of composability (functions which lock should not call other functions which lock, even if the mutexes are different)
  - Risk of deadlock
  - High overhead (requires system call, which modifies kernel data structures)
  - Lack of scalability caused by coarse-grained locking
  - Code complexity and increased overhead caused by fine-grained locking

Lock-free Programming

- What is meant by a lock-free program?
  - A lock-free program is a multi-threaded program which does not rely on locking and unlocking mutexes
- Briefly list some of the advantages of lock-free programming
  - If done correctly, threads can never block each other
  - No possibility of deadlock or livelock
  - If a thread is blocked, other threads can continue to execute
  - Useful if work must be completed within a time limit (e.g. real time systems)

Lock-free Programming

- Briefly list some of the disadvantages of lock-free programming
  - Programs can be hard to understand, with very subtle bugs
  - Can be very difficult to write correct, efficient lock-free code and data structures
  - For many applications, the extra complexity is often not justified