CS4021/4521 Advanced Computer Architecture II

Dr Jeremy Jones


some changes from previous years
- parallel algorithms on shared memory multiprocessors
- maximise performance
- DNA analysis (bioinformatics, string searching)
- multiple threads, SIMD instructions (SSEx, AVXx, ...)
- cost of sharing data, memory bandwidth, the problem with locks, ... 
- lock implementations
- CAS based lockless algorithms (for lists, trees, hash tables...)
- lockless algorithms using hardware transactional memory
Remember

• algorithms underpin most of what Computer Scientists and Engineers do

• "algorithms are the poetry of the 21st century"
human genome approx. 3.2 billion bases (length n)

ACGTACGTGGTAACCCGGTTA.....

DNA sequencing techniques generate millions of short reads of 100 or so bases (length m)

CGTGGTAA...

have to find the reads in the genome (maybe with inexact matching)

using the Burroughs Wheeler Transform (BWT) can find each read in $O(m)$ rather than $O(n)$

can parallelise BWT generation and searching for the reads
Consider a Binary Search Tree

- add (50)
- add(45)
- remove(45) – NO children
- remove(25) – ONE child
- remove(20) – TWO children

find node (20)
find smallest key in its right sub tree (22)
overwrite key 20 with 22
remove old node 22 (will have zero or one child)
Concurrent add operations

• concurrently add(27) and add(50)
  
  OK if adding to different leaf nodes

• concurrently add(23) and add(24)
  
  problem as adding to same leaf node
  
  result depends on how the steps of the operations are interleaved
  
  could work correctly, BUT...
  
  if there's a conflict ONLY one node may be added

• concurrent removes also possible if there is no conflict
Concurrent Updates

- binary search tree normally protected by a single lock so concurrent updates by multiple threads are NOT possible

- plenty of scope for concurrent updates provided they do NOT conflict with one other (with a large tree conflicts will be rare)

- hence lockless algorithms
• lock based algorithms are pessimistic
  - assumes something will always go wrong

• lockless algorithms are optimistic
  - assumes nothing will go wrong, deal with conflicts when detected
  - exploits parallelism
  - higher throughput / performance if done correctly

• lockless algorithms implemented using CAS instructions of hardware transactional memory (Intel TSX instruction set)
LOOKING FORWARD TO SEEING YOU IN SEPTEMBER