Read The Art of Multiprocessor Programming by Maurice Herlihy & Nir Shavit

Read several articles on Locklessinc.com/articles

Including Spinlocks and Read Write locks

23/01/14

Agreed with David Gregg to implement a Ring Buffer as the first data structure. I decided to go for a simple design in C to begin with. I implemented it using pthreads and two modes of operation; the first used mutexes in a simple lock and unlock fashion while the second used a spinlock. However, the implementation of the spinlock proved tricky due to the low level nature of C and so I decided early on to start using C++, in order to access the higher level intrinsics, since my focus was not on the implementation but rather the testing of these structures.

<http://people.csail.mit.edu/edya/publications/OptimisticFIFOQueue-DISC2004.pdf> --Gives details on implementing a buffer using CAS.

27/01/14

Researched several papers on the topic of concurrent data structures, placed them into the research folder

Amended the output of the program to be easier to paste into excel for graph generation

Gathered data from stoker for both locked and spinlock modes of ring buffer

Need to gather data from my machine, spoon and ducss

28/01/14

Gathered data from ducss, netsoc and local machine. VPN has only 1 core so no point testing on that. Tested assembly spinlock against C++ version, performance negligible.

29/01/14

Added in several references to the report

30/01/14

Added in a CAS lock with back off to the ring buffer and collected data from all 4 machines

02/02/14

Added a TAS and ticket lock to all 4 architectures and collected data

03/02/14

Got perf + nice working. Came to the conclusion that a graph should be done for each size and mode of operation possible. In addition, started work on a linked list.

06/02/14

Added to the ring buffer section of the report, gathered data on the CAS lock with no back off.

11/02/14

Implemented an assembly version of the ticket lock to the ring buffer to compare against the c++ ticket lock. The difference was negligible.

Started gathering data from stoker using perf

Started gathering data from cube, perf not installed not, have requested

12/02/14

Gathered data on the ring buffer from cube using perf. In addition I …

…implemented a spsc lockless ring buffer and gathered data on it from stoker, cube and my local machine both normally and with perf

13/02/14

Finished off the locked version of the linked list and messed around …

…was CAS in test.cpp

17/02/14

Added modes of operation TAS with pause and TTAS with no pause for ri…

…ng buffer, collected data on stoker,local and cube

18/02/14

[Finished locked linked list, add modes of operation and gathered data…](https://github.com/gibsonma/FYP/commit/2a2e7cf234a6634fe4b9357f2ffda65a83fa0de9) […](https://github.com/gibsonma/FYP/commits/master)

… from stoker, local and cube. Added CASLOCKND which is a CAS lock but with no delay. In addition, I ran tests to determine if locked performance differed with different key ranges. In most cases it does not, though with a very small key range such as 10, progress slows drastically.

19/02/14

Added code for a lockless linked list using compare and swap and gathered data for three key ranges, 100, 100000 and 100000000 from stoker, cube and local.

22/02/14

Added to the report

25/02/14

Created graphs for ring buffer and linked list both locked and lockless

26/02/14

Gathered data from linked list using perf

01/03/14

Implemented four more modes into linked list, TTAS\_RELAX, TAS\_RELAX, CASLOCK\_RELAX & TICKET\_RELAX which all use \_mm\_pause() instead of sleep. Note that CASLOCK does well and TICKET is now working as expected.

Did the same for locked Ring buffer, not lockless and added the data to the sheet

04/03/14

Implemented MPMC, a lockless linked list, with a head and a tail. Nodes are constantly added onto the head and removed from the tail. This is to eliminate the time used for a list to be traversed to find the point of entry. I also created its own data table. Stoker currently busy but will use averages to get data if necessary.