**Description of Algorithms**

Bucket sort algorithm with various types of locks. This cope helps to evaluate the performance of various locks with bucket sort. The algorithm description about bucket sort is said in lab1 .

Tas- Test and Set lock. No fairness + Bad performance

TTAS - Test and Test and Set lock . No fairness + Good performance

Ticket lock - Not good performance. Fairness.

MCS - Good Performance + Fairness

Pthread - Sleeping mutex lock.

**A brief description of your code organization**

**Fucntions present in locks.c, locks.h**

**Acquire\_ticketlock** - function to acquire lock in Ticket

**Release\_ticketlock** - Function to release lock in ticket

**TAS\_lock** - TAS lock acquire

**Test\_TAS\_lock** - TTAS lock acquire

**Unlock** - unlcok function for TAS and TTAS

**Pthread\_lock** - mutex lock acquire

**Pthread\_unlock** - mutex lock release

**Pthread\_bar** - pthread barrier wait

**Sense\_bar** - sense barrier wait

**Mcs\_acquire** - Mcs lock acquire in class MCS

**Mcs\_release** - MCS lock release in class MCS

**Class** **MCS\_node** - node structure for MCS lock

**A description of every file submitted**

**Source:**

**counter.cpp**

Counter program with locks and barriers

**locks.cpp**

Code for all locks and barriers

**main.cpp**

Main program which calls bucket sort algorithm and reads input elements from source file

**sort\_algorithm\_bucket.cpp**

Bucket sort algorithm

**Sort\_algorithm\_fj.cpp**

Fork join algorithm (but not used for lab 2)

**Includes**

**Locks.h**

Header file for the functions of locks and barriers

**Main.h**

Header file for main program of sorting

**Sort\_algorithm\_fj.h**

Header files for fork join sort (not used in lab 2)

**sort\_algorithm\_bucket.h**

Header files for bucket sort algorithm

**Compilation instructions**

Make - generates executable for bucket sort code

Make counter - generates executable for counter program

**Execution instructions**

**./counter -t [number of thread] -i [no of iterations] --lock=<tas, ttas, ticket, mcs, pthread> --bar=<sense,pthread> -o <output file>**

eg:

./counter -t 50 -i 1000 --lock=ttas -o output\_counter.txt

./counter -t 50 -i 1000 --bar=sense -o output\_counter.txt

**./mysort [source filename] -o [output filename] -t [num of threads] --algs=bucket --lock=<tas,ttas,ticket, mcs, pthread>**

./mysort input.txt -o steve.txt -t 15 --algs=bucket --lock=tas

**Any extant bugs**

Run time of MCS and sense reversal barrier is found to be very long. I guess further work has to be done to optimize run time

Counter program

Threads = 50

Iterations = 10

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Lock** | **Run time from perf**  **(secs)** | **Run time**  **(secs)** | **L1 cache hit rate** | **Branch prediction hit rate** | **Page fault count** |
| TAS | 0.0060 | 0.000293 | 88.26% | 97.1% | 230 |
| TTAS | 0.0019 | 0.000370 | 95.4% | 97.69% | 233 |
| Ticket | 0.0018 | 0.000264 | 94.71% | 97.75% | 232 |
| MCS | 0.0018 | 0.000386 | 94.5% | 97.53 | 231 |
| pthread | 0.0063 | 0.000458 | 87.47% | 96.94% | 231 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Barrier** | **Run time from perf**  **(secs)** | **Run time**  **(secs)** | **L1 cache hit rate** | **Branch prediction hit rate** | **Page fault count** |
| Pthread barrier | 0.032 | 0.0264 | 88.35% | 98.13% | 234 |
| Sense reversal | 17.26 | 17.26 | 99.9975% | 99.9989% | 241 |

**Result Analysis of locks:**

TTAS is found to have the highest hit rate because it spins on checking the globals before writing to the globals which is in contrast to TAS lock where it writes first and due to all copies of cache are invalidated and miss rate increases in TAS.

Ticket lock has a lower cache hit because cache invalidation occurs frequently because it increments the global turn variable.

MCS is found to have a very good cache hit when compared to TAS, pthread as the cache misses will be very less. Cache misses are very less in MCS because the locks spins on the local node rather than a global variable. This improves cache performance as well as it is more preferred than TTAS (though it has high cache hit) because it is FIFO lock thereby gives good fairness.

The run time of pthread mutex lock is found to be very high because mutex sleeps and it has to do lots of context switches. The difference between mutex and other spin lock is that spinlock doesn’t sleep whereas mutex lock sleeps.

**Result Analysis of barriers:**

Sense reversal barrier has a better cache hit rate because they spin on local sense variable than a global as that of pthread barrier

Bucket sort program

Threads = 15

Elements = 1555

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Lock** | **Run time from perf**  **(secs)** | **Run time**  **(secs)** | **L1 cache hit rate** | **Branch prediction hit rate** | **Page fault count** |
| TAS | 0.0028 | 0.001488 | 99.09% | 98.77% | 192 |
| TTAS | 0.00312 | 0.001429 | 99.09% | 98.94% | 191 |
| Ticket | 0.00256 | 0.001064 | 98.65% | 99.65% | 188 |
| MCS | 0.006713 | 0.001301 | 98.61% | 99.62% | 197 |
| pthread | 0.003153 | 0.001668 | 96.27% | 97.94% | 190 |