

18-344 Recitation 9

Lab 5 - Synchronization: Overview

Task 1: Giga-Updates Per Second (GUPS) Kernel

NoSync:

```
for(int i = 0; i < num_iters; i++){  
    size_t ind = rand() % kv_entries;  
    kv[ind]++;  
}
```

KV	1	0	1	2	2	2	1	1	10th iteration
KV	1	0	1	2	3	2	1	1	11th iteration

Task 1: GUPS Kernel

Parameters:

`num_entries` - Number of entries in the Key - Value array

`num_threads` - Number of threads running a kernel

`num_iters` - number of times each thread will run the kernel

Task 1: GUPS Kernel

Fix synchronization problems using:

1. `pthread_spinlock_t`
2. `__sync_fetch_and_add`
3. `__sync_bool_compare_and_swap`
4. Transactional Memory

All fixes should have the same sum!

Task 2: Swap Within Array Partition Set (SWAPS) Kernel

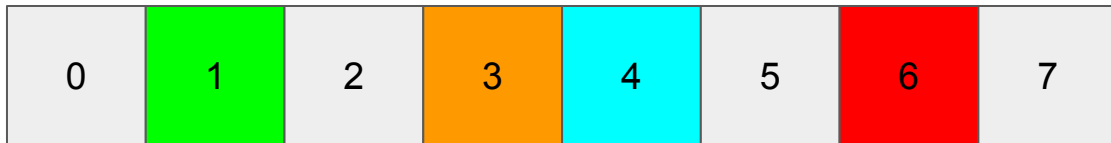
```
for(int i = 0; i < num_iters; i++){
    uniq_list(kv_entries, num_swaps, inds);
    for(int j = 0; j < num_swaps; j++){
        int j2 = j == 0 ? (num_swaps-1) : j-1;
        unsigned long t = kv[ inds[j2] ];
        kv[ inds[j2] ] = kv[ inds[j] ];
        kv[ inds[j] ] = t+1;
    }
}
```

inds

6	3	1	4
---	---	---	---

SWAPS Kernel

KV



nth iteration

6 <-> 4

KV



3 <-> 6

KV



1 <-> 3

KV



4 <-> 1

KV



KV



n+1 th iteration

Task 2: SWAPS Kernel

Parameters:

num_entries - Number of entries in the Key - Value array

num_threads - Number of threads running a kernel

num_iters - number of times each thread will run the kernel

num_swaps - number of swaps per kernel execution

Intel Transactional Memory

General Psuedo-code

```
for(i = 0..MAX_TRIES) {  
    if(xbegin()) {  
        ...; xend();  
        goto done;  
    }  
}  
  
//Fallback code here  
Lock()  
... // do work  
Unlock()  
  
Done:  
    //continue
```

Reasons a transaction might abort

- Too many blocks with their TM bits set leaves no room for more TM blocks
- Too many defined as “more blocks w/ TM bits set than blocks in a way”
- Conflict with another transaction or non-transactional access
- identified through incoming coherence traffic
- Explicit xabort() instruction when transactional code concludes transaction is not useful
- Other, unspecified, but arbitrary conditions left up to the micro architects
- I speculate that these are related to internal buffers of fixed capacity

Lab Logistics

You will be using Intel Transactional Memory, make sure to use a machine that supports it

`ece029`, `ece030`, `ece031`

You can double check the TSX is enabled using the binary:

```
./has-tsx
```

Useful resource

[A Primer on Memory Consistency and Cache Coherence](#)

- Daniel J. Sorin Mark D. Hill David A. Wood