CSCI 4061 Discussion 10

4/2/18

Overview

- More Synchronization
  - Condition variables
  - Semaphores
- PA3

More Synchronization

- What if we need to wait for something other than a critical section?

- Example:
  - Waiting for a job queue to fill up

Spin Locks

- Have a thread constantly check a variable to see if a condition is true.

```
while (flag == false) {} // Spin until true.
```

- What are the issues with this?

Condition Variables

- Method of blocking while waiting for a condition to be satisfied.
  - A thread blocks itself when the condition is false.
  - A different thread wakes the first one when the condition is true.

Condition Variable Code

```
 pthread_cond_t cond;                            // Create a cond variable.
 pthread_cond_init(pthread_cond_t* cond);    // Initialize a cond variable.

// Wait on a condition variable.
pthread_cond_wait(pthread_cond_t* cond, pthread_mutex_t* mutex);

// Signal one or all (broadcast) threads waiting on cond variable.
pthread_cond_signal(pthread_cond_t* cond);
pthread_cond_broadcast(pthread_cond_t* cond);
```

Example

```
pthread_mutex_lock(&m);
while (...some condition is false...)
   pthread_cond_wait(&c, &m);

... critical section ...

pthread_mutex_unlock(&m);
```

Spurious Wake-ups

Place the call to pthread_cond_wait inside of a **while loop** so that if the thread is signaled before the condition is true (spurious wake-up) the thread will go back to blocking.

Semaphores

- A lock with a counter added.
- ‘Posting’ to a semaphore increases the count.
- ‘Polling’ on a semaphore decreases the count.
    - If the count is 0, block until a post occurs.

Semaphore Functions

```
// Create a semaphore.
sem_t semaphore;

// Initialize a semaphore.
sem_init(sem_t* sem);

sem_wait(sem_t* sem); // Wait on/decrement the counter of a semaphore.
sem_post(sem_t* sem); // Increment the counter of the semaphore.
```

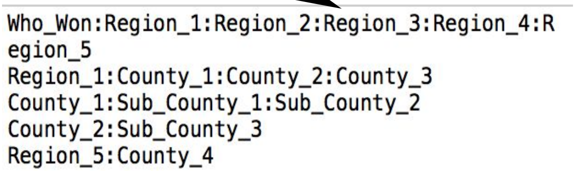
Exercise

- You will synchronize a shared queue between producer and consumer threads, first using condition variables and then using semaphores.
- You may **only** alter functions with TODO in the comment above them.
- Your solutions are timed and cannot run slower than twice that of the solution.
- Code to get started is provided in rec10.c

PA3

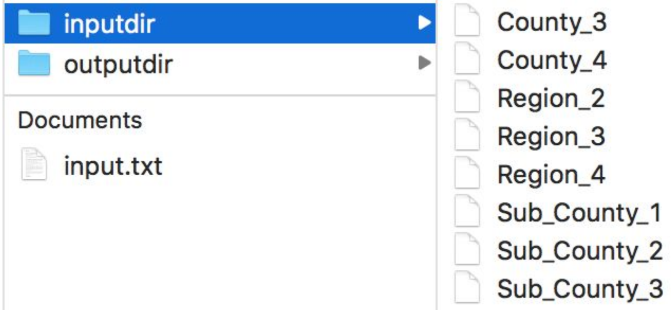
```
./votecounter <DAG.txt> <inputdir> <outputdir>
```

Given

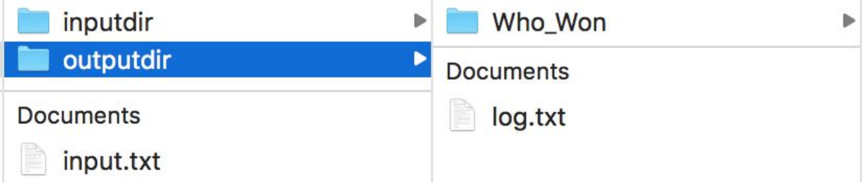


```
Who_Won:Region_1:Region_2:Region_3:Region_4:R
egion_5
Region_1:County_1:County_2:County_3
County_1:Sub_County_1:Sub_County_2
County_2:Sub_County_3
Region_5:County_4
```

Given



You need to create this directory



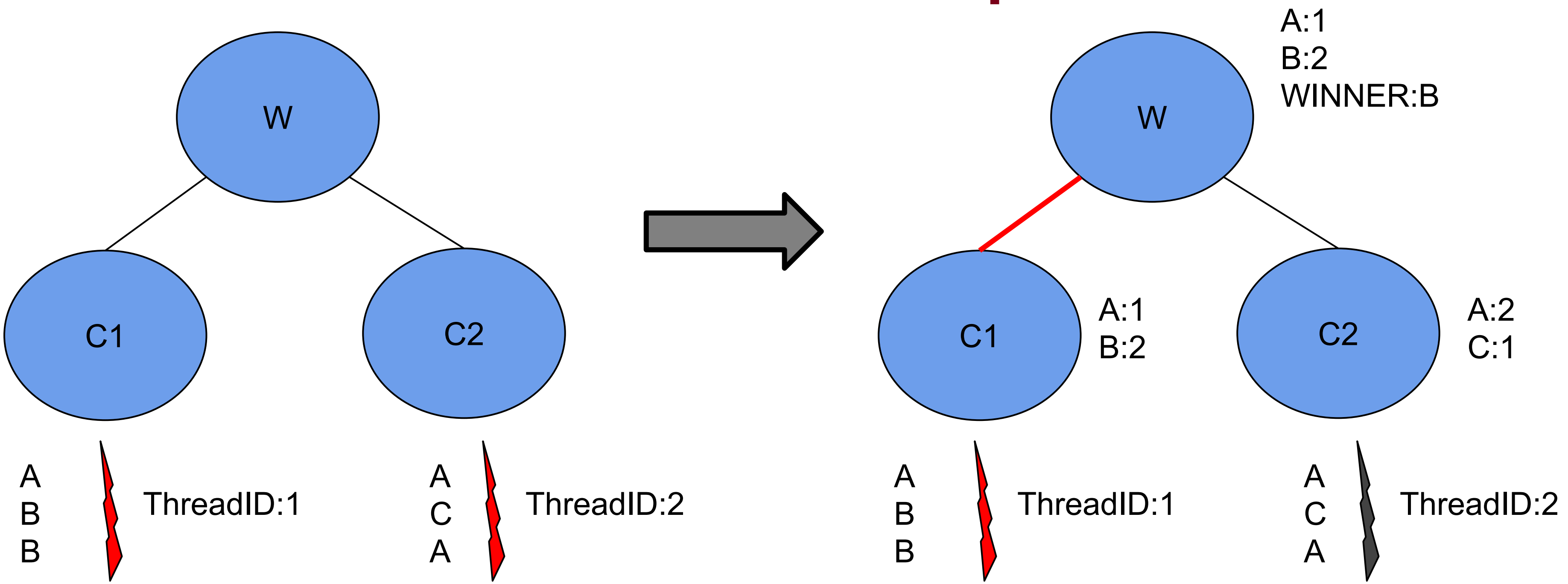
Note that the log.txt should be under output dir.

PA3 - Main Idea

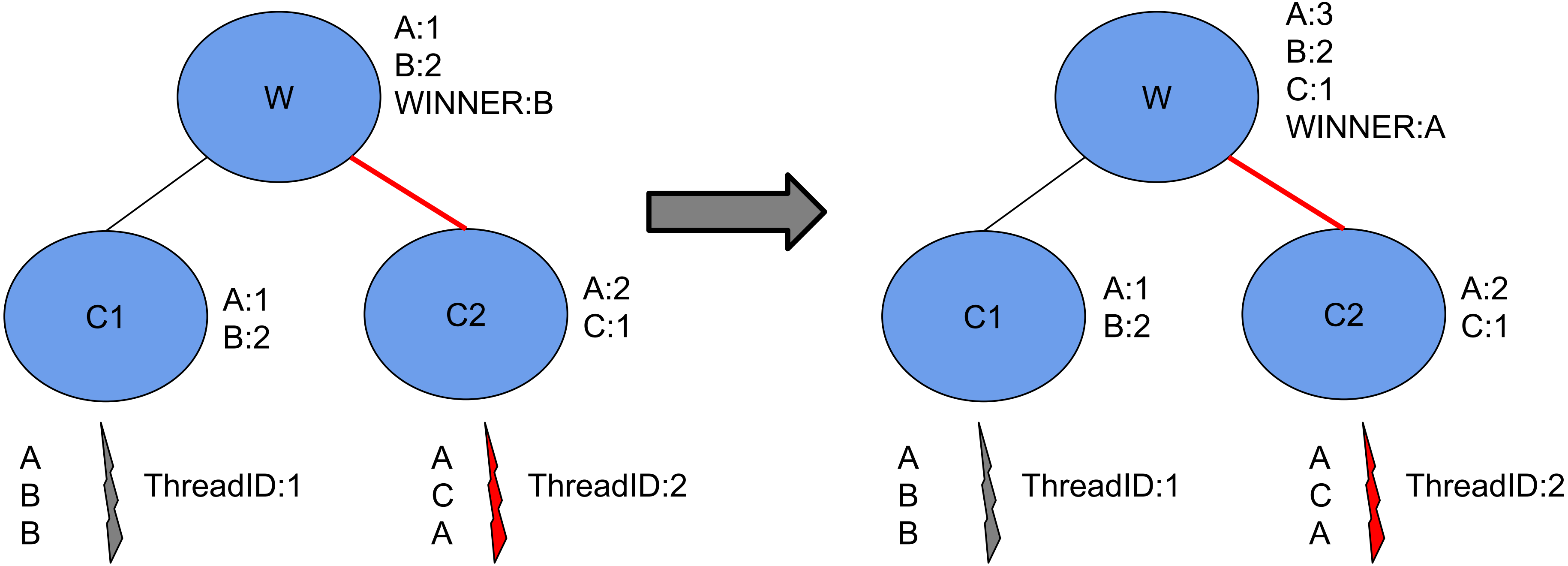
```
./votecounter <DAG.txt> <inputdir> <outputdir>
```

- Read <DAG.txt> and create <outputdir>.
- Start <number_threads> equal to number of files in <inputdir>, Where each thread:
  - Decrypt a file inside <inputdir>, and place it in the right place under <outputdir>.
  - Aggregate recursively from the leaf node until the thread reach the root and declare the winner.

PA3 - Example


W
C1
C2
A
B
B
ThreadID:1
A
C
A
ThreadID:2
A:1
B:2
WINNER:B
W
C1
A:1
B:2
C2
A:2
C:1
A
B
B
ThreadID:1
A
C
A
ThreadID:2

PA3 - Example


W
A:1
B:2
WINNER:B
C1
A:1
B:2
C2
A:2
C:1
A
B
B
ThreadID:1
A
C
A
ThreadID:2
W
A:3
B:2
C:1
WINNER:A
C1
A:1
B:2
C2
A:2
C:1
A
B
B
ThreadID:1
A
C
A
ThreadID:2