Multicore Programming Project 2

-Simple Multiversion Concurrency Control (MVCC)-

Due Date : 11:59 pm October 7 (Wed), 2015

Specifications

- Each thread has two variables ***A and B*** initialized with random values.
- A thread should maintain multi-versioned values for these.
  - (a, b, thread version)→(a, b, thread version)→(a, b, thread version) …
- The sum of A and B is constant: ***A+B = Constant***.
  - The sum of all pairs of (A, B) of all threads is also constant.
- Only the owner thread can modify its A and B while other threads can read them.
- The version of values each thread can read from other thread's variables is determined by a thread's read-view.
- A ***read-view*** is a kind of a snapshot that determines which version of values each thread can safely read.
  - A thread can read a version of (A, B, version k) if a version *k* is the most up-to-date version which was created before the thread started.

Operation Logic (UPDATE operation)

- A thread X increments and get a global execution order (k) and inserts its order (k) into the global active thread list. Then it creates its read-view that has to capture a list of active threads (**ATOMICALLY**).
- Execute following sequence:
  - Pick a thread *i* randomly.
  - Read correct version of (*Ai*, *Bi)* from *i* using its read-view.
  - $Ax = Ax + Ai$ and $Bx = Bx - Ai$.
  - ***Verify the invariant : Constant =*** $\sum_{i \in Thread}(Ai + Bi)$ ***(based on its read-view)***
  - Create a new version (Ax, Bx, k).
- Remove thread execution order (k) from the global active thread list (**ATOMICALLY**).

Constraints

- The mutual exclusion should be guaranteed by your own lock object, i.e., ***Bakery algorithm with your own optimization***. Compare this with the implementation using ***pthread_mutex***().

- You MUST use a plain "***singly linked list***" for maintaining a globally shared active thread list.

- You MUST use a plain "***singly linked list***" for maintaining SingleWriterMultipleReader multi-versioned pairs of (A, B).

- Old version of pair needs to be removed after you guarantee that no other thread would need that version. (garbage-collected).
- Input arguments: ***--num_thread --duration***

Goals

- Measure/maximize the total throughput (t) of all threads : $\sum updates/sec$.
- If you improve fairness, that's a plus point:

$$\mathcal{J}(x_1, x_2, \ldots, x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \cdot \sum_{i=1}^{n} {x_i}^2}$$

- Your program should print both ***throughput*** and ***fairness*** at the end of its execution.