1.) Without any lock unlock calls the global variable is modified simultaneously by the 3 different threads, the local count is not. Because of this the global variable “count” is inaccurately summed when threads are running concurrently to avoid this overwriting the threads locked before another thread can modify the count.
2.) With a smaller loop bound there is no inconsistency in the final value because the scheduler can complete the threads sums before moving to the next thread, when the scheduler is dealing with larger tasks it jumps between the threads allowing for the inconsistency seen in problem 1.
3.) Local variables are always consistent because they are local, the threads are not modifying a shared memory space when modifying the local variables.
4.) By guardian against writes from other threads using locking a thread is allowed to finish modifying the global variable before the next thread begins modifying it ensuring a consistent sum every time.
5.) The discrepancy in the time is a result of the scheduler having to wait for one thread to finish before implementing the task in the other thread. Without the locks the scheduler is allowed to more quickly implement the program, albeit giving the incorrect result.