Chris Bernosky

1. There are many things that may be responsible for the inconsistency of the final count from the actual value to the expected value. The main thing though is the fact that the lock takes longer to run because it only runs one thread at a time. When the lock was taken off of the code it ran in a considerably less amount of time, but was much less reliable.
2. The big reason there is little inconsistency is simply because the loop is smaller, as stated in the question, proving less room for error. A larger loop creates timing iterations that may not be as consistent since there is much more going on in a larger loop than in a smaller one. This is why locks are necessary to ensure that quality is not being compensated for time by modifying multiple threads at a time.
3. Local variables are not modified by multiple threads, since they are consistently one thread that receives no changes during the process of the program they will always be much more consistent than a global variable.
4. The lock allows for only one thread to be open and modified at the time. Then this was used in the code, it ran slower with a count of 300,000,000 but was much more reliable. The code without the lock modifies multiple threads at a time, allowing a faster runtime, but much less reliability.
5. As stated above, the lock took much longer, almost twice the amount of time, locking all other threads and only allowing one to be modified at a time. The unlocked code took much less time because it does not utilize the lock and unlock features, allowing it to modify all open threads at the same time.