6a. Why is threading useful on a single-core processor?

It is useful in saving processor time and utilization.

- In a single core processor system, only one thread can be active at any time. Suppose one youtube thread is waiting for the data from network, then the scheduler can switch to another thread that is not waiting.

It improves the responsiveness to users

- Depending on the thread priority and OS, threads executes one after another after a particular allocation time or switching done by OS. But it gives us an illustration that they are running simultaneously, relative to the user applications.

6b. Do more threads always mean better performance?

We opt threading for parallelizing a task. Running more threads than the CPU supports is actually serializing but not parallelizing. In reality, running too many threads can slow down a program. First, running many threads requires partitioning the task into smaller subtasks and it involves starting, swapping and terminating threads. Second, running many threads involves the cost of sharing hardware resources. So increase in number of threads does not increase the performance.

6c. Is super-linear speedup possible? Explain why or why not.

Super-linear speedup means that speedup is greater than the number of processors that are used. It is possible in low level computations because of the cache effect. Cache effect deriving from different memory hierarchies of a modern computer, in low level computations is the one reason for super-linear speedup. In parallel computing, the numbers of processes change along with the size of accumulated caches from different processors. With greater accumulated cache size, almost all or a bit less working set can cached, and the memory access time reduces dramatically, which results the extra speedup in addition to that from the actual computation.

6d. Why are locks needed in a multi-threaded program?

Ans: The locks are mainly needed for restricting the access to a resource where there are numerous threads in execution. It provides programmatic control and also highly flexible in terms of usage of locks as they can be accessed as well as released in any manner without any specifications in general.

6e. Would it make sense to limit the number of threads in a server process?

Ans:

It is better to limit the number of threads in a server process.

Threads needs memory to setup their own private stack. As a result to serve efficiently, threads use too much memory. In addition to this, independent threads always act in a disorganized way. This leads to page thrashing, as it is difficult to build a stable working set in a virtual memory system. In such cases, there is a depletion in page performance. Suppose let's consider everything fits into the memory, still we can observe the chaotic pattern in rendering caches. So again, performance decreases compared to a single threaded system.