1. How do you determine the number of threads you should launch to realistically test your application?

Base it on the number of cores. If 4 cores (and each hyperthreaded) then can have a max of 8 threads dedicated to your app.
To test your app, check low load, typical load, full load and then full load plus. Low maybe 2 threads, typical depends on the threaded tasks you have, high would be 1 thread per core or 8 threads for above configuration, and 8 plus to test full load with waiting threads.

You should determine your machines capacity at runtime. For C++ 11 the following call works

```
unsigned concurentThreadsSupported = std::thread::hardware_concurrency();
```

and base the number of threads you launch based on this number.

You should also consider if threads are CPU intensive or IO intensive. If IO intensive you can have more threads running since most of the time they will be blocked.

2. If you are launching several threads, under what circumstances would these threads not run in parallel on a modern processor?

Anytime a multiple threads are waiting to enter the same critical section then they run in serial, not parrallel.

An egregious case is when a thread defines a critical section to be the entirety of its work, so it acquires a sync object and then completes this work before exiting. All other threads waiting to enter that critical section are blocked until the thread owning the synchronization object finishes and releases the object. This poor coding practice makes a solution less efficient than a single threaded process due to the overhead associated with creating and tearing down threads.

Now if you launch many more threads than you have cores what happens is you will have a max of number cores threads running while the rest wait in a ready or blocked queue. But they will be swapped in as the currently running threads are swapped out. They will not all be running in parrallel, but the ones on the cores are.