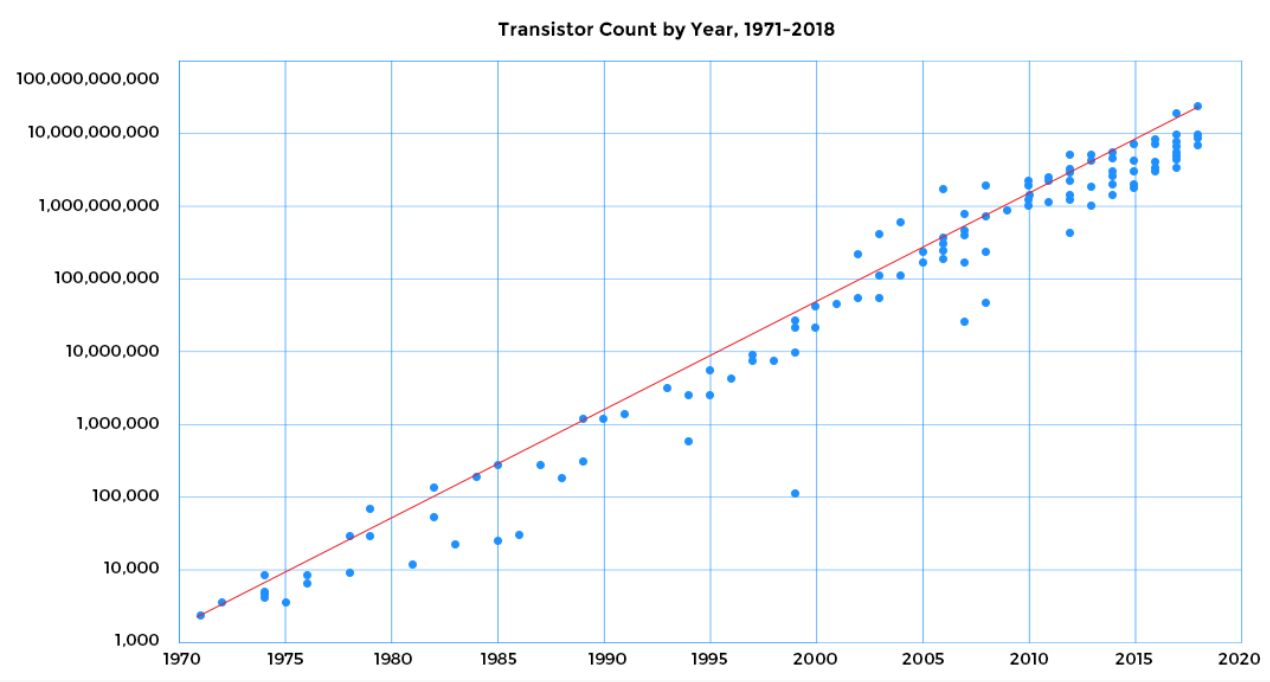
Moore's Law

Discusses impact of Moore's law on concurrency.

Moore's Law

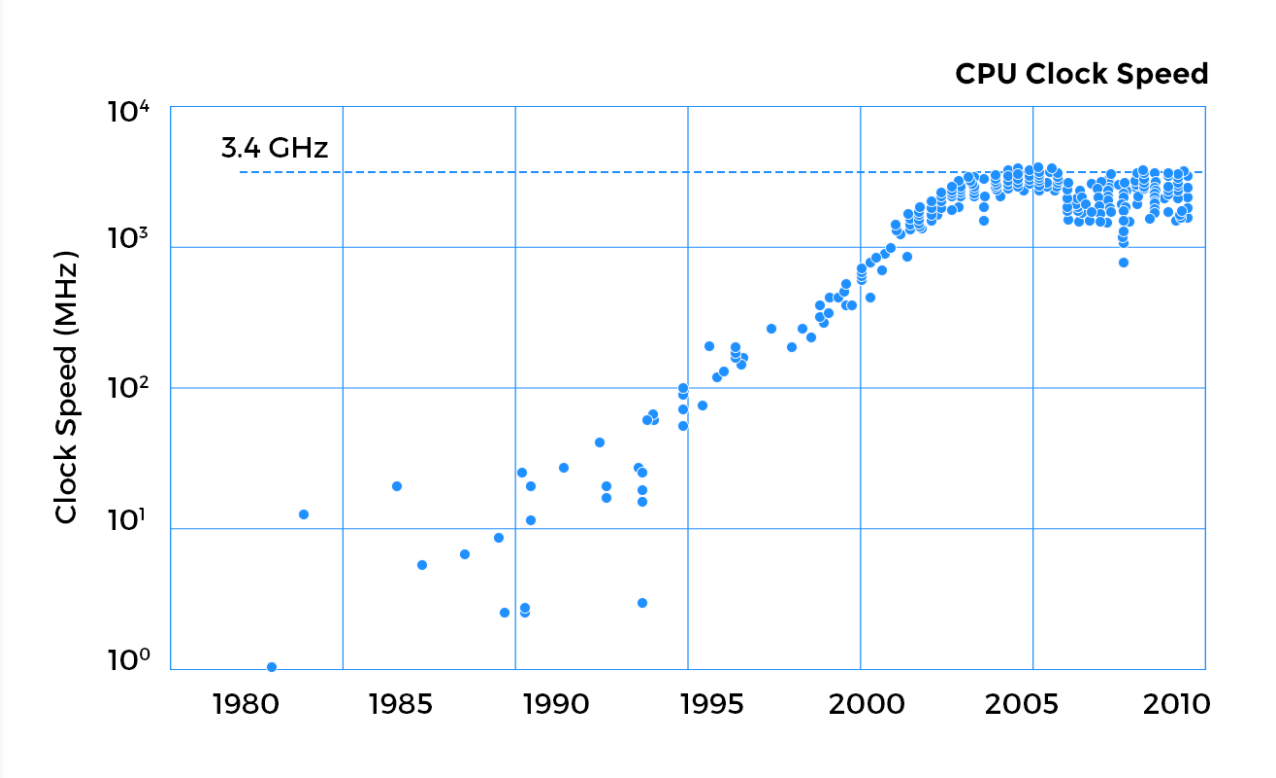
In this lesson, we'll cover a high-level overview of Moore's law to present a final motivation to the reader for writing multi-threaded applications and realize that concurrency is inevitable in the future of software.

Gordon Moore, co-founder of Intel, observed the number of transistors that can be packed into a given unit of space doubles about every two years and in turn the processing power of computers doubles and the cost halves. Moore's law is more of an observation than a law grounded in formal scientific research. It states that **the number of transistors per square inch on a chip will double every two years**. This exponential growth has been going on since the 70's and is only now starting to slow down. The following graph shows the growth of the transistor count.



Initially, the clock speeds of processors also doubled along with the transistor count. This is because as transistors get smaller, their

frequency increases and propagation delays decrease because now the transistors are packed closer together. However, the promise of exponential growth by Moore's law came to an end more than a decade ago with respect to clock speeds. The increase in clock speeds of processors has slowed down much faster than the increase in number of transistors that can be placed on a microchip. If we plot clock speeds we find that the linear exponential growth stopped after 2003 and the trend line flattened out. The clock speed (proportional to difference between supply voltage and threshold voltage) cannot increase because the supply voltage is already down to an extent where it cannot be decreased to get dramatic gains in clock speed. **In 10 years from 2000 to 2009, clock speed just increased from 1.3 GHz to 2.8 GHz merely doubling in a decade rather than increasing 32 times as expected by Moore's law.** The following plot shows the clock speeds flattening out towards 2010.



Since processors aren't getting faster as quickly as they use to, we need alternative measures to achieve performance gains. One of the ways to do that is to use multicore processors. Introduced in the early 2000s, multicore processors have more than one CPU on the same machine. To exploit this processing power, programs must be written as multi-threaded applications. A single-threaded application running on an octa-core processor will only use 1/8th of the total throughput of that machine, which is unacceptable in most scenarios.

Another analogy is to think of a bullock cart being pulled by an ox. We

Another analogy is to think of a bullock cart being pulled by an ox. We can breed the ox to be stronger and more powerful to pull more load but

eventually there's a limit to how strong the ox can get. To pull more load, an easier solution is to attach several oxen to the bullock cart. The computing industry is also going in the direction of this analogy.