Moore’s Law

By Stephen Kawaguchi

Moore’s Law is named after Gordon Moore, the co-founder of Intel. It is not actually a law of physics or any sort, but rather an observation.

Moore predicted that transistor density would double every two years. With increasing density came increased speed. As a result, chips went through a long period where speeds would drastically increase every few months and double approximately every two years – this was essentially an exponential growth in speed over time.

Eventually, Moore’s Law hit its limit we began to hit the limits of physics.

First, as transistors become denser, they consume more power. As power consumption increases, temperature increases and more heat is generated. There are physical limits as to how much heat we can feasibly manage, especially in mobile and portable devices. Air cooling is the most practical, and so the limits of what we can do has halted Moore’s Law.

A second limiting factor is Dennard Scaling. As we scale voltage this reduces dynamic power consumption, and thus heat generation. The problems arises because there is a limit as to how low we can go before it becomes unreliable to detect transistor state switches. Essentially, voltage scaling can’t prevent power leakage loss, especially since as transistors become smaller, their insulation becomes thinner and thinner. This leakage results in noise – in other words, the threshold voltage we must rely on to detect switches has to stay large enough to be reliable. So this limits the amount of voltage scaling that we can actually do.

So, in short, we have ultimately hit the physical limitations of transistor size due to power and heat to where Moore’s Law is no longer true. We can still see improvements in transistors, but not at the rate that we saw for the decades after Moore made his observation.

Today, instead of making smaller transistors – and thus more powerful CPUs, we make multi-core processors. This has led to an important shift where programmers have to learn to deal with concurrency rather than being able to count on CPU clock speeds increasing over time. This in turn has given rise to cloud computing, where we use many thousands of commodity (cheap) computers to process our programs instead of relying on a single machine or even a single CPU. This is why languages like Go and Node have gained in popularity because they make concurrency easier to deal with.