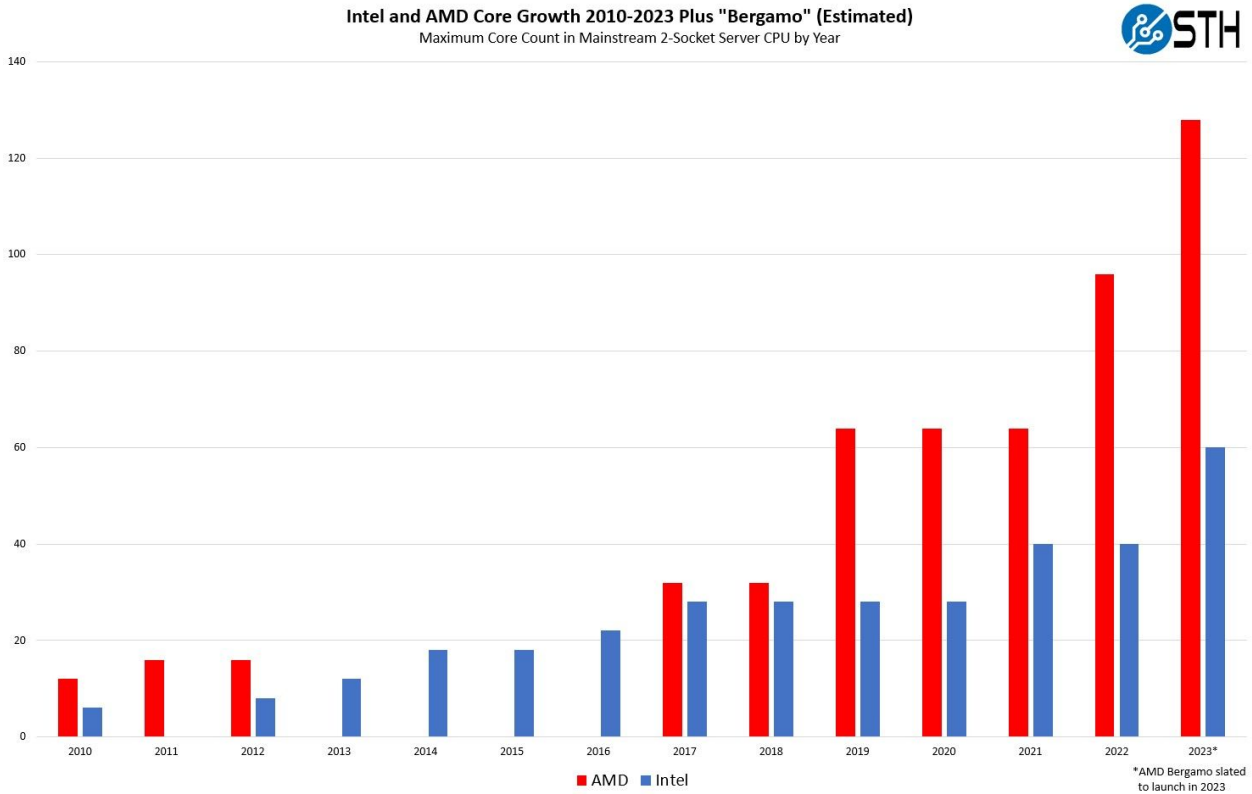
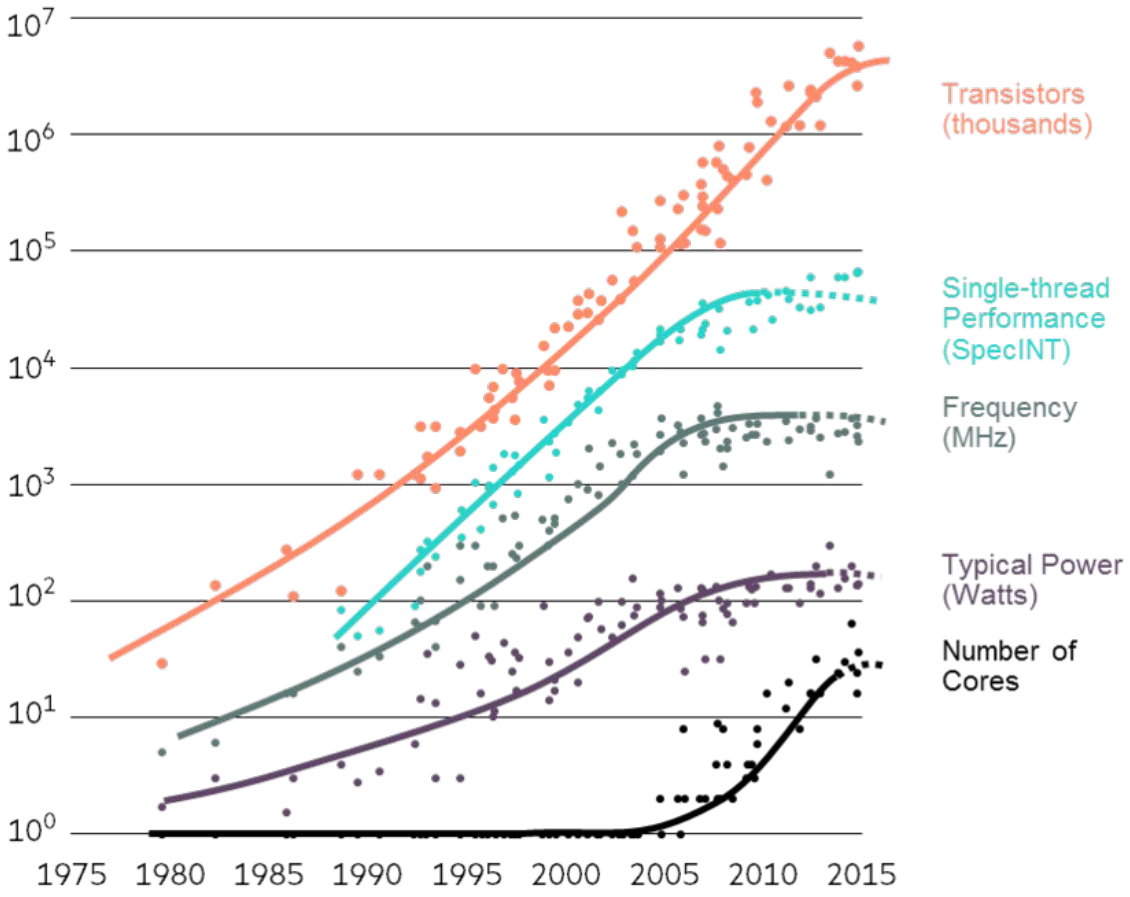
julia +

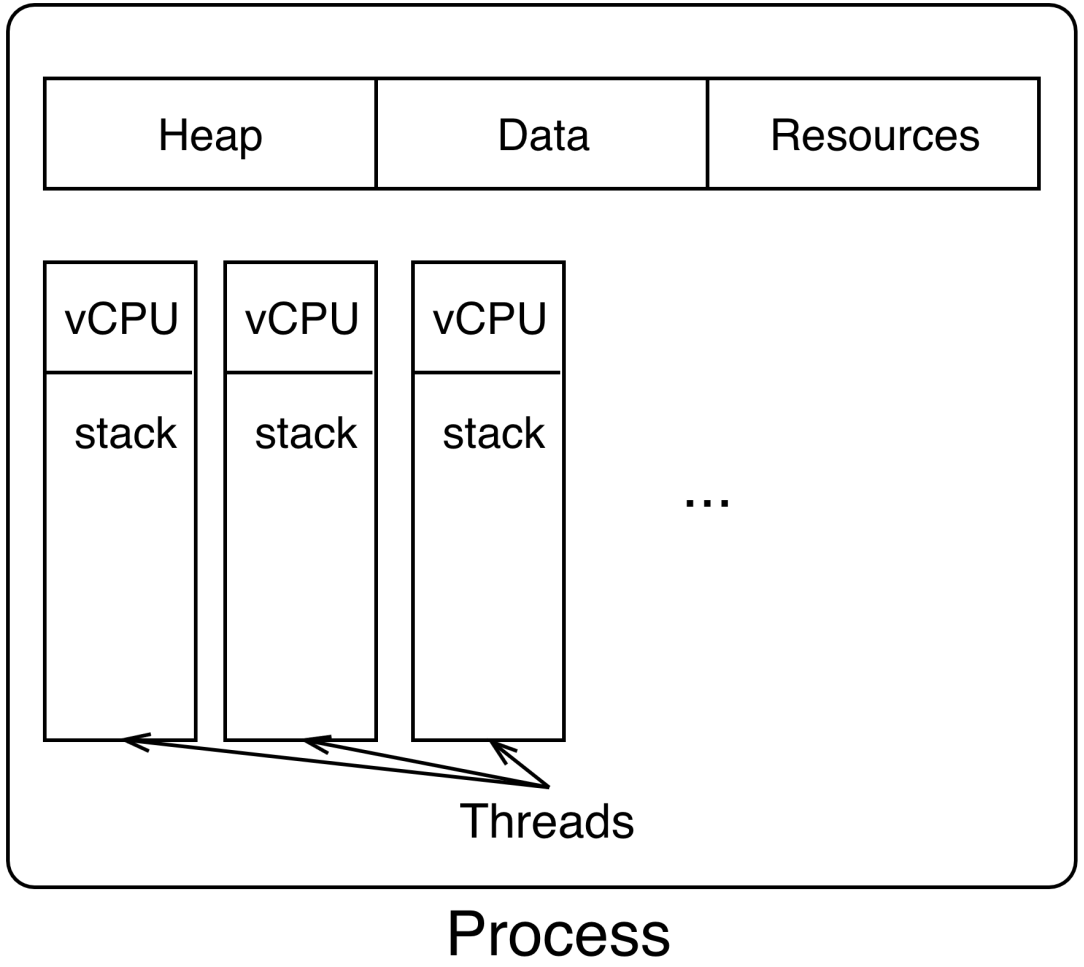
Threading is the basis for modern CPU Speed



Microprocessors



- **Process = Heavyweight**
- **Thread = Lightweight**
- Threads share heap
- Don’t share stack
- Most programs have 1 process and n threads



Thread vs Task vs Core in Julia

A **Task** in Julia is the **unit of work**, and is executed on a thread. It may be interrupted and move to a new thread by Julia!

A **Thread** in Julia is a virtualization of a CPU core. Julia executes a task on one thread out of a **threadpool**.

A **Core** is a physical resource, one of several “CPU”s on the processor in your computer.

The Great Naming Mistake

```
function foo(x)
    Threads.@threads for i ∈ eachindex(x)
        println("Thread $(Threads.threadid()) processing element $i")
    end
end
```

What does this code do?

@spawn vs @threads in Julia

- @threads is a relatively old API, often limited to :static scheduling
  - Static scheduling interferes with Julia's ability to interleave different tasks. It's **not cooperative**
- @spawn involves a lot more bookkeeping for simple tasks like parallelizing a for loop
- @spawn **is cooperative and much more general** (any function)

Can we have the best of both worlds?