Here are some tricks for understanding the performance of parallel software

Remember:
Always, always, always try the simplest parallel solution first, then measure performance to see where you stand.

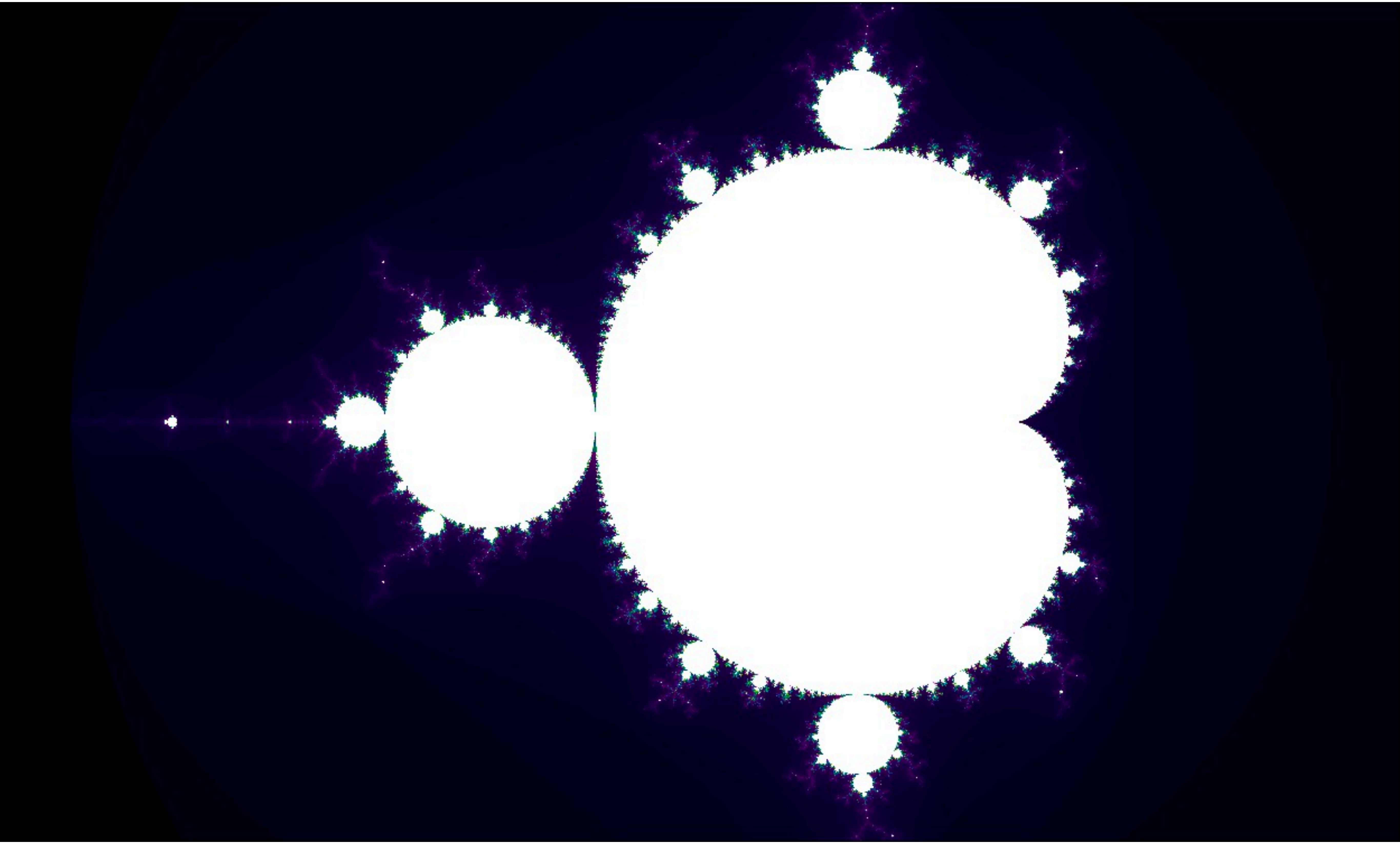
A useful performance analysis strategy

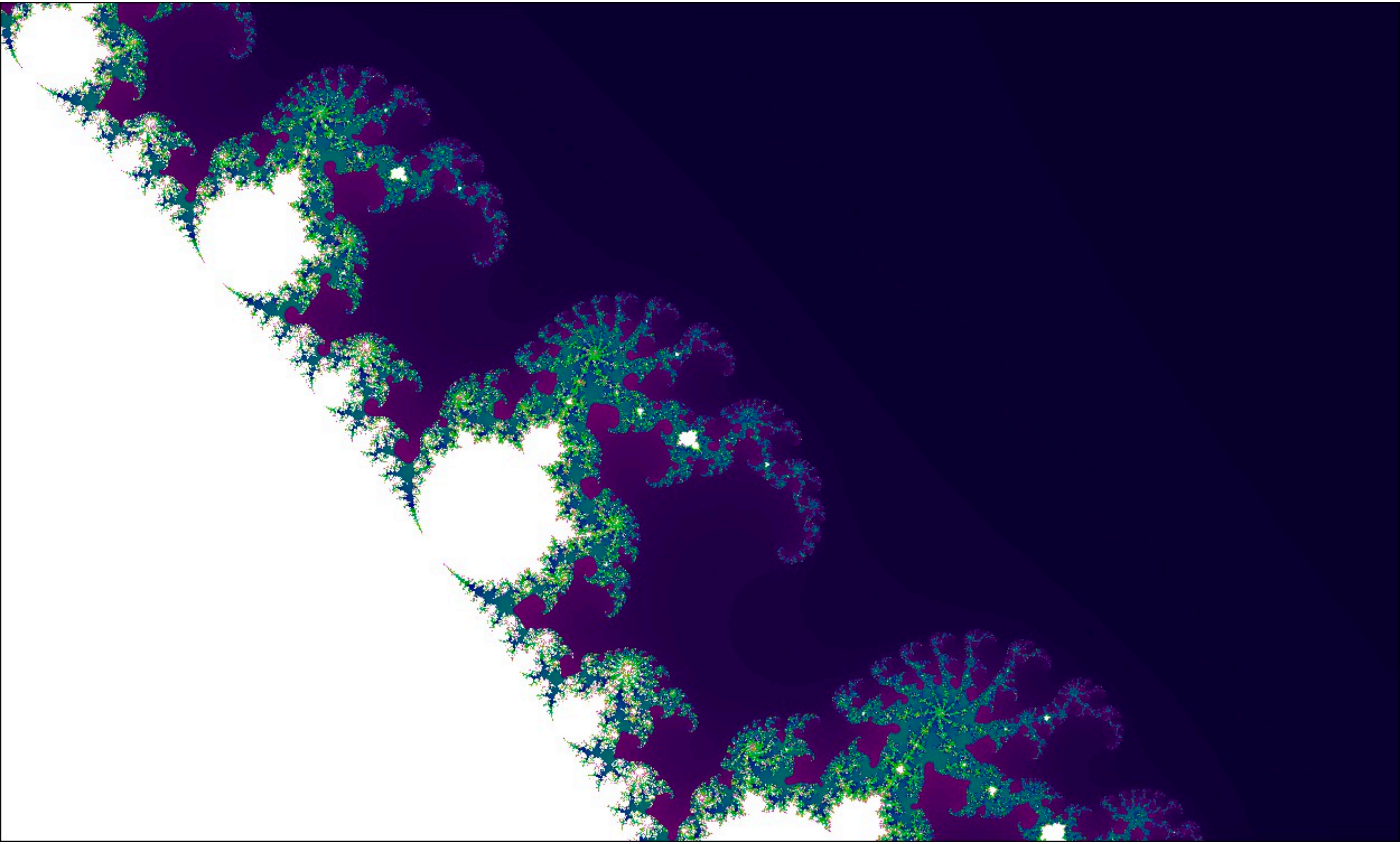
- **Determine if your performance is limited by computation, imbalance, memory bandwidth (or memory latency), or synchronization?**

- **Try and establish “high watermarks”**
  - **What’s the best you can do in practice?**
  - **How close is your implementation to a best-case scenario?**

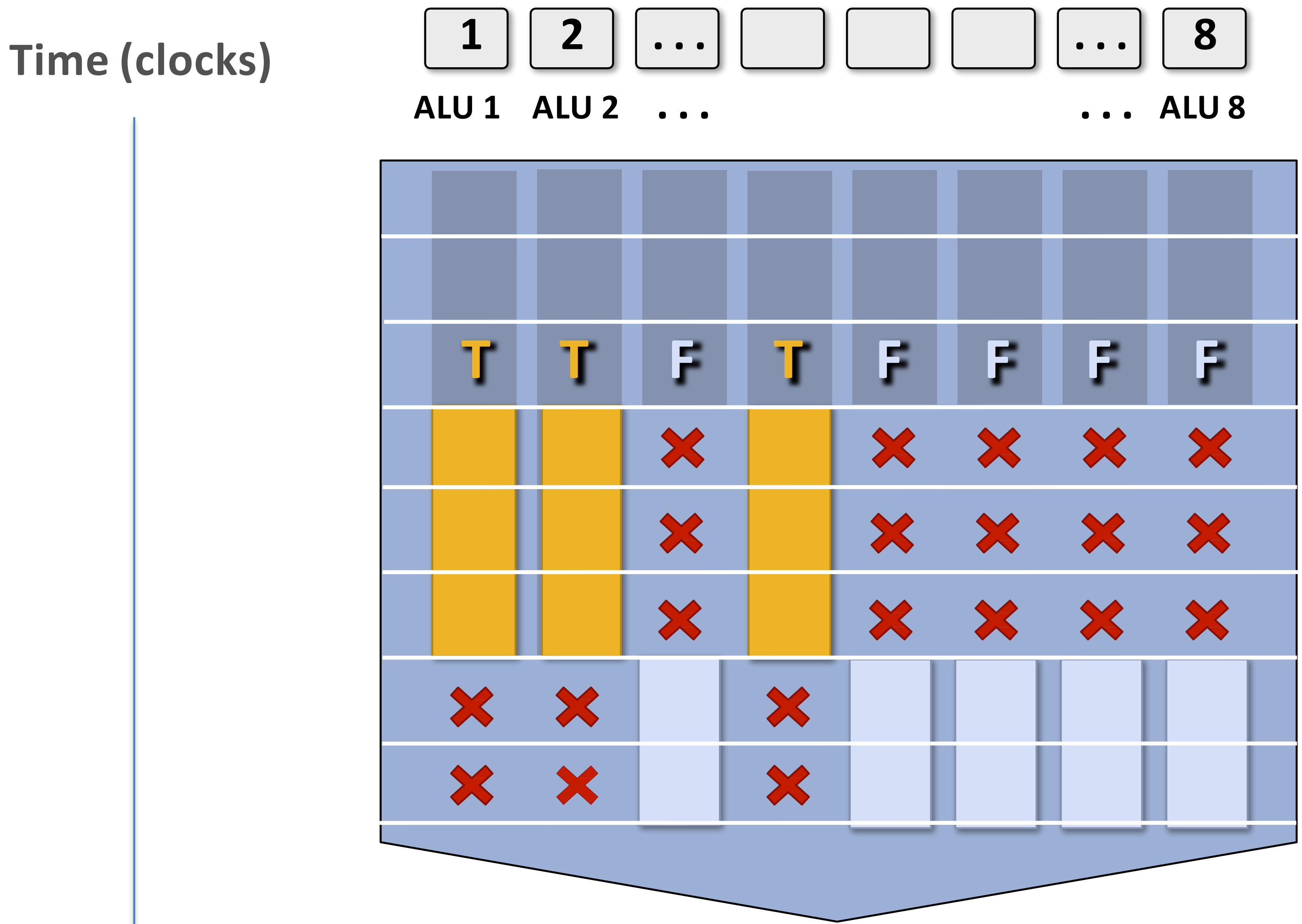
$$sin\ x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} \cdots$$

$$cos\ x = x - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} \cdots$$





Mask (discard) output of ALU



(assume logic below is to be executed for each element in input array 'A' producing output into the array 'result')

```
static inline int mandel(..) {
    float z_re = c_re, z_im = c_im;
    int i;
    for (i = 0; i < count; ++i) {

        if (z_re * z_re + z_im * z_im > 4.f)
           break;

        float new_re = z_re*z_re - z_im*z_im;
        float new_im = 2.f * z_re * z_im;
        z_re = c_re + new_re;
        z_im = c_im + new_im;
    }

    return i;
}
```

Not all ALUs do useful work!

Worst case: 1/8 peak performance