SPECTRAL:
TECHNOLOGIES

WEEK 7

Time measurement

Time measurement is tricky

- The measurement has some limit to it's precision
- The measurement itself may introduce overhead
- There is a difference between taking a timestamp and converting it to the actual time moment and measuring elapsed time
- There are different types of elapsed time
  - wall time
  - cpu time
  - user time
  - system time

Wall vs Cpu time

- Lets assume we run some process or function and want to measure the elapsed time
- *Wall time* is the real elapsed time, measured by wall clock (*real* stat reported by *time* utility)
- *Cpu time* is the time, cpu spent executing the function. Waiting for I/O isn't included here. If the process is multithreaded, then the contribution of all threads is summed
- *User time* is the portion of cpu time calculated in user mode only
- *Sys time* is the portion of cpu time calculated in kernel mode only

Monotonic clocks vs real-time clocks

- Real-time (or wall) clock is a clock, thats tied with real time. Such clock is usually synchronized with global time and can be adjusted by NTP or by user manually
  - Thats why its not good for measuring elapsed time
  - But it can be used for getting the actual current time
- Monotonic clock is a clock, that provides constant time intervals between it's ticks
  - So its better for measuring elapsed time

C++ time functions

- `std::chrono::system_clock`
- `std::chrono::steady_clock`
- `clock`
- `clock_gettime`
- `time`
- `gettimeofday`
- `rdtsc` (cycles)

std::chrono clocks

- `std::chrono::system_clock` is a wall clock
  - uses either `clock_gettime`, `gettimeofday` or `time` under the hood
  - most likely `clock_gettime(CLOCK_REALTIME)` in linux libstdc++
- `std::chrono::steady_clock` is a monotonic clock
  - Uses `clock_gettime(CLOCK_MONOTONIC)` under the hood
- `std::chrono::high_resolution_clock` is a synonym for one of these clocks, so don't use it

C functions

- `clock()` returns cpu time
  - Uses `clock_gettime` under the hood currently
- `time` returns real time
  - Uses `sys_time` syscall internally
  - 1s precision, so its not interesting
- `gettimeofday` returns real time
  - considered obsolete in a favor of `clock_gettime`

clock_gettime

- The entry point for both cpu and wall clocks plus some variants of those
- nanosecond precision
- latency may vary from 20 to 100 ns (but usually close to 20)

rdtsc

- Reads the current value of the processor's time-stamp counter (TSC)
- Its not easy to convert results to the elapsed time since the actual cpu frequency isn't constant
- Different cores provide different values
- Out-of-order execution can mess with results, cpuid can be used
- Costs about 20 cycles, so the latency is less than 10ns
- Use `__rdtsc` to call
- Can be used for performance comparison in ulta low-latency applications

Summing up

- Use `std::chrono::system_clock` to get the real time
- `std::chrono::steady_clock` to get the elapsed time
- `clock_gettime(CLOCK_MONOTONIC_RAW)` may be more accurate
- `rdtsc` when the measured block takes nanoseconds to execute