CPU Load Experiments

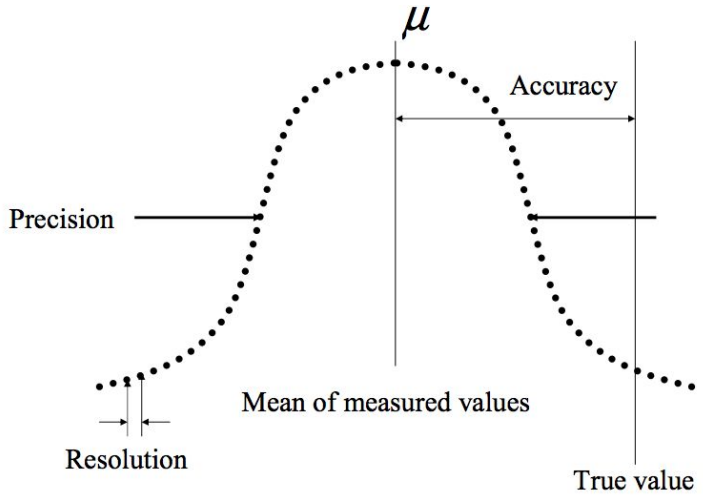
David Herrera
Aug 3th/2017

Sources of error

- Random Error
  - Unpredictable (At least very hard to control)
  - Non-deterministic
  - Unbiased
  - E.g. Noise in a copper wire telephone communication, anything that can be approximated with a Gaussian of mean zero
  - Avoided by statistical analysis
- Systematic Error
  - Result of an experimental “mistake”
  - Avoided by control through skill of experimenter
  - Produces a constant or slowly varying biased
  - E.g. Forgetting to clean cache

More on Errors

- Systematic errors are unfairly called ERRORS, some of them are rather overlooks by experimenter. (Mytkowicz et al.)
- Holistic conclusions made about the system based on a small sample of local data or in computer science terms, a very specific experimental setup.



What source of error is the CPU Load?

What source of error is the CPU Load?

1. Random Error
    a. Hard to control or predict processes
    b. The OS is a very complicated intricate system with many moving components.
2. Systematic Error
    a. Well, we can control some aspect of it. How?
    b. There are two solutions:
        i. Turn off wifi connections, kill any process taking a lot of CPU, get into the system deamons and kill some of those.
        ii. Completely characterize CPU load. I.e. For each benchmark, know the run-time for 50%, 70%, >50% but less than 70% etc.
            1. Requires having a system to control CPU load at a given point of time
            2. Also assumes ideally a complete control of the CPU load
            3. With these, results can be published where we would think of each CPU Load percentage as a different experimental setup

Let's take a step back

What is the CPU load?

- CPU has certain, limited processing power
  - Processes are scheduled by priority, to ensure each process is treated fairly, they are given a "quantum" of time to run, before being placed back in the run queue.
  - Two Times: user time, system time.

How do we calculate it?

```
static computeMeanCpu(start,end) {
    //Calculate the difference in idle and total time between the measures
    var idleDifference = end.idle - start.idle;
    var totalDifference = end.total - start.total;
    //Calculate the average percentage CPU usage
    var percentageCPU = 100 - (100 * idleDifference / totalDifference);
    return Number(percentageCPU.toFixed(2));
}
```

The Experiment...

What do we hope to achieve?

- Determine if there is a correlation between CPU load and running time.
- Integrate it into wu-wei, perhaps as a sort of flag to the experimenter.
- Learn a lesson and have this present whenever we benchmark.

Procedure to get good results - First Try

- **Idea**
  - Measure two points before/after benchmark, take average from that
  - To get measurements of different CPU load levels, continue along with my daily activities.
- **Problem**
  - Two points is not enough to characterize CPU, might have a case where the CPU varies a lot while the benchmark is running, (specially since I am taking the approach of randomly varying it)
  - CPU load is too sensitive, right before and right after might not be a good characterization with wu wei running

Procedure to get good results - Second Try

- **Idea**
  - Make COMPLICATED system of promises, measure a few points before, a few points after, and during the benchmark execution
  - Again, to gather measurements of different CPU load levels, continue along with my daily activities as benchmarks are running
- **Improvements**
  - A little bit more data!!!
  - Characterization of actual benchmark CPU load is beginning to form
- **Problems**
  - Did not distinguished between before, during and after points, assumption was that the actual benchmark would not induce a very high cost on the CPU load
  - As with the previous approach, the assumption of “continuing with daily life” let to bad characterizations of CPU-load, the longer the runtime of the benchmark, the more likely fore the measurement to be incorrect.

Procedure to get good results - Third Try

- **Idea**
  - Collect CPU load at all times during experiment
  - No need for promises, open synchronous processes that hold the execution of wu-wei for 5 secs, before and after benchmark
  - Collect data also during the benchmark in order to characterize benchmark CPU load
  - Still collect data at random.
- **Improvements**
  - This data was pretty good, gave a pretty good idea of CPU load vs. time,
  - Gave a decent and noticeable enough characterization of benchmark.
- **Problems**
  - Not all benchmarks had data for certain CPU-loads, did not allow to make complete conclusions about all benchmarks
  - Still not controlling the CPU-load introduces a systematic error due to the fact that averages are computed

Procedure to get good results - Finally...

- CPU traces from previous trail give a pretty good idea of CPU load per benchmark
- **Idea**
  - Well, now that we know a good estimate of how much CPU load a given benchmark induces, use one of them to control and drive CPU load. Therefore, use benchmark as a background process to raise and lower CPU.
  - No need to randomly collect points anymore, turn off, wifi, kill other processes, make sure to control as much of the system in that sense as possible.
  - Perform experiments at 40%, 60%, and 80% CPU-load.
- **Results**
  - Wallah!!!