Overview

Measuring performance & benchmarking

References:
- Section 3.2 in: Chellappa, Franchetti, Püschel: How To Write Fast Numerical Code: A Small Introduction, GTTSE 2008
Benchmarking

First: Validate/test your code!

Measure runtime (in [s] or [cycles]) for a set of relevant input sizes

- **seconds**: actual runtime
- **cycles**: abstracts from CPU frequency

Usually: Compute and show performance (in [flop/s] or [flop/cycle])

*Careful*: Better performance ≠ better runtime (why?)

- Op count could differ
- Never show in one plot performance of two algorithms with substantially different op count

---

How to Measure Runtime?

**C clock()**

- process specific, low resolution, very portable

**gettimeofday**

- measures wall clock time, higher resolution, somewhat portable

Performance counter (e.g., TSC on Intel)

- measures cycles (i.e., also wall clock time), highest resolution, not portable

*Careful*:

- measure only what you want to measure
- ensure proper machine state
  (e.g., cold or warm cache = input data is or is not in cache)
- measure enough repetitions
- check how reproducible; if not reproducible: fix it

*Getting proper measurements is not easy at all!*
Problems with Timing

Too few iterations: inaccurate non-reproducible timing
Too many iterations: system events interfere
Machine is under load: produces side effects
Multiple timings performed on the same machine

Bad data alignment of input/output vectors:
- align to multiples of cache line (on Core: address is divisible by 64)
- sometimes aligning to page boundaries (address divisible by 4096) makes sense

Machine was not rebooted for a long time: state of operating system causes problems
Computation is input data dependent: choose representative input data
Computation is inplace and data grows until an exception is triggered (computation is done with NaNs)
You work on a computer that has dynamic frequency scaling (e.g., turbo boost)

Always check whether timings make sense, are reproducible

Benchmarks in Writing

Specify experimental setup
- platform
- compiler and version
- compiler flags used

Plot: Very readable
- Title, x-label, y-label should be there
- Fonts large enough
- Enough contrast (e.g., no yellow on white please)
- Proper number format
  - No: 13.254687; yes: 13.25
  - No: 2.0345e-05 s; yes: 20.3 μs
  - No: 100000 B; maybe: 100,000 B; yes: 100 KB
What’s Suboptimal?

- Ugly font
- Fully saturated color
- Legends cause long decoding time
- Grid lines compete with data lines (poor layering)
- What's Suboptimal?

Performance of DFT $2^n$ on Pentium 4, 2.53 GHz

- Spiral SSE
- Intel MKL interl.
- Spiral C
- Spiral C vect

Gflop/s

log$_2$(input size)

$\log_2(n)$

Horizontal y-label

No y-axis (superfluous)

Left alignment

Attractive font (sans serif, avoid Arial)
Calibri, Helvetica, Gill Sans MT, ...

Main line possibly emphasized (red, thicker)

Background/grid inverted for better layering

No legend; makes decoding easier

http://funnyimagewebsite.blogspot.ch/