Benchmarking Numerical Code

Markus Püschel

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Benchmarking

- ***First:*** **Verify 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!***

Example: Timing MMM

- Assume `MMM(A,B,C,n)` computes
  `C = C + AB`, `A,B,C` are `nxn` matrices

```
double time_MMM(int n)
{ // allocate
  double *A=(double*)malloc(n*n*sizeof(double));
  double *B=(double*)malloc(n*n*sizeof(double));
  double *C=(double*)malloc(n*n*sizeof(double));

  // initialize
  for (int i = 0; i < n*n; i++){
    A[i] = B[i] = C[i] = 0.0;
  }

  init_MMM(A,B,C,n); // if needed

  // warm up cache (for warm cache timing)
  MMM(A,B,C,n);

  // time
  ReadTime(t0);
  for (int i = 0; i < TIMING_REPETITIONS; i++)
    MMM(A,B,C,n);
  ReadTime(t1);

  // compute runtime
  return (double)((t1-t0)/TIMING_REPETITIONS);
}
```

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)
- Time stamp counter (if used) overflows
- 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 laptop that has dynamic frequency scaling
- *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 (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*
- ***How to make a decent plot?***

Left alignment

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

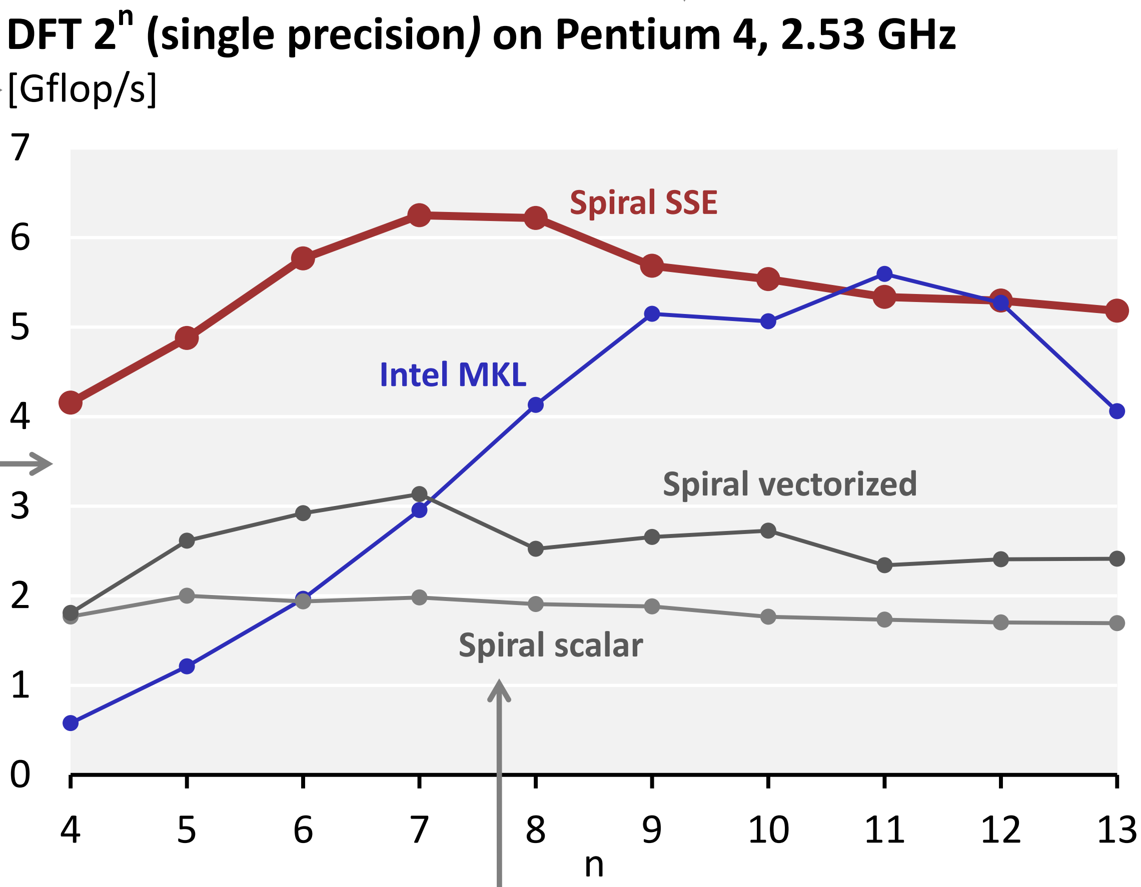
Horizontal y-label

No y-axis (superfluous)

Main line possibly emphasized (red, thicker)

Background/grid inverted for better layering

No legend; makes decoding easier


DFT 2^n (single precision) on Pentium 4, 2.53 GHz
[Gflop/s]
Spiral SSE
Intel MKL
Spiral vectorized
Spiral scalar
n