Advanced Systems Lab
Spring 2023
Lecture: Cost analysis and performance

Instructor: Markus Püschel, Ce Zhang
TA: Joao Rivera, several more

Organization

Team and research project: Deadline: March 10th
If you need team: fastcode-forum@lists.inf.ethz.ch
Matrix-Matrix Multiplication (MMM) on 2 x Core 2 Duo 3 GHz

Performance [Gflop/s]

- Compiler doesn’t do the job
- Doing by hand: nightmare

Performance is different than other software quality features
Today

Asymptotic analysis
Cost analysis and performance


Asymptotic Analysis of Algorithms

Analysis for
- Runtime
- Space (= memory footprint)
- Data movement (e.g., between cache and memory)

Example MMM: C = A*B + C, A,B,C are all n x n
- Runtime: $O(n^3)$
- Space: $O(n^3)$
Valid?

Is asymptotic analysis still valid given this?

All algorithms are $O(n^3)$ when counting flops.

What happens to asymptotics if I take memory accesses into account?
No problem: $O(f(n))$ flops means at most $O(f(n))$ memory accesses

What happens if I take vectorization/parallelization into account?
More parameters needed: E.g., $O(n^3/p)$ on $p$ processors

Asymptotic Analysis: Limitations

$\Theta(f(n))$ describes only the eventual trend of the runtime

Constants matter
- Not clear when “eventual” starts
- $n^2$ is likely better than $1000n^2$
- $10000000000n$ is likely worse than $n^2$
Cost Analysis for Numerical Problems

Goal: determine exact “cost” of an algorithm

Cost = number of relevant operations

Formally: define cost measure $C(n)$. Examples:

- Counting adds and mults separately: $C(n) = (\text{adds}(n), \text{mults}(n))$
- Counting adds, mults, divs separately: $C(n) = (\text{adds}(n), \text{mults}(n), \text{divs}(n))$
- Counting all flops together: $C(n) = \text{flops}(n)$

This course: focus on floating point operations

The cost measure usually counts only the operations that constitute the mathematical algorithm (e.g., as written on paper) and not operations that arise due to its mapping on a computer (e.g., index computations, data movement).

Example: next slide

Example

```c
/* Multiply n x n matrices a and b */
void mmm(double *a, double *b, double *c, int n) {
    int i, j, k;
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            for (k = 0; k < n; k++)
                c[i*n + j] += a[i*n + k]*b[k*n + j];
}
```

Asymptotic runtime

- $O(n^3)$

Cost measure?

- $C(n) = (\text{fladds}(n), \text{flmults}(n)) = (n^3, n^3)$
- $C(n) = \text{flops}(n) = 2n^3$
Cost Analysis: How To Do

Define suitable cost measure

Count in algorithm or code
  - Recursive function: solve recurrence

Instrument code

Use performance counters
  - Intel PCM
  - Intel Vtune
  - Perfmon (open source)
  - Counters for floating points are recently less and less available

Remember: Even Exact Cost ≠ Runtime

Matrix-Matrix Multiplication (MMM) on 2 x Core 2 Duo 3 GHz

Performance [Gflop/s]

<table>
<thead>
<tr>
<th>Matrix size</th>
<th>0</th>
<th>1,000</th>
<th>2,000</th>
<th>3,000</th>
<th>4,000</th>
<th>5,000</th>
<th>6,000</th>
<th>7,000</th>
<th>8,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>flops</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>25</td>
<td>30</td>
<td>35</td>
<td>40</td>
<td>45</td>
</tr>
</tbody>
</table>

2n³ flops
Why Cost Analysis?

Enables performance analysis:

\[
\text{performance} = \frac{\text{cost}}{\text{runtime}} \quad \text{[flops/cycle] or [flops/sec]}
\]

Upper bound through machine’s peak performance

Example

/* Matrix-vector multiplication y = Ax + y */
void mmm(double *A, double *x, double *y, int n) {
    int i, j, k;
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            y[i] += A[i*n + j]*x[j];
}

Flops? For \( n = 10 \):

- \( 2n^2, 200 \)

Performance for \( n = 10 \) if runs in 400 cycles

- 0.5 flops/cycle

Assume peak performance: 2 flops/cycle

percentage peak?

- 25%
Summary

Asymptotic runtime gives only an idea of the runtime \textit{trend}

Exact number of operations (cost):
- Also no good indicator of runtime
- But enables performance analysis
- Upper bound on performance through computer’s peak performance = lower bound on achievable runtime

Always measure performance (if possible)
- Gives idea of efficiency
- Gives percentage of peak

In the project, a performance plot is one of the first milestones