B4M36ESW: Efficient software
Lecture 2: Benchmarking

Michal Sojka
sojkam1@fel.cvut.cz

February 27, 2017
Wikipedia defines benchmark as:

1. the act of running a computer program, a set of programs, or other operations, in order to **assess the relative performance** of an **object**, normally by running a number of standard tests and trials against it.

2. a benchmarking program

Object examples:
- Hardware
- Compiler
- Algorithm
- ...

Types of benchmarks:
- Micro-benchmarks (synthetic)
- Application benchmarks
How to measure software performance?

- What to measure?
  - Execution time
  - Memory consumption
  - Energy

Not as easy as it sounds

See the rest of the lecture
How to measure software performance?

- What to measure?
  - Execution time
  - Memory consumption
  - Energy

- How to measure?
  - Not as easy as it sounds
  - See the rest of the lecture
Measuring energy

- Connect power meter to your computer/board
- Use hardware-provided interfaces for power/energy measurement/control
Measuring energy

- Connect power meter to your computer/board
- Use hardware-provided interfaces for power/energy measurement/control

Intel RAPL (Running Average Power Limit)

- Allows to monitor and/or limit power consumption of individual components
- Package domain, memory domain (DRAM)
- Interface via MSRs
Measuring memory consumption

- Program memory (code, static data, heap, stack)
  - Stack is allocated for each thread
- Operating system memory
  - Allocated by OS kernel on behalf of the program
  - Network buffers, disk and file system caches, system objects (timers, semaphores, …)
- Shared libraries
Measuring execution time

Outline

1. Measuring execution time
   - Repeating iterations
   - Repeating executions
   - Repeating compilation
   - Multi-level repetition

2. Measuring speedup
Measuring execution time

Timestamping

- Use system calls
  - Linux: gettimeofday, clock_gettime(CLOCK_MONITONIC)
  - Overhead – hundreds of cycles
  - Optimization: Virtual syscall
Measuring execution time

Timestamping

- Use system calls
  - Linux: gettimeofday, clock_gettime(CLOCK_MONITONIC)
  - Overhead – hundreds of cycles
  - Optimization: Virtual syscall

- Use hardware directly (timestamp counter)

```c
static inline uint64_t rdtsc()
{
    uint64_t ret;
    asm volatile ( "rdtsc" : "=A"(ret) );
    return ret;
}
```
Execution time exhibits variations
Influenced by many factors
- Hardware, input data, compiler, memory layout, measuring overhead, rest of the system, network load, ... you name it
- Same factors can be controlled, others cannot

Repeatability of measurements
How to design benchmark experiments properly?
How to measure speedup?
Example
Measuring execution time

The Challenge of Reasonable Repetition

- Variations
- Measurements must be repeated
- We want to eliminate the influence of random (non-deterministic) factors

Statistics

- Controlled variables (e.g. compiler flags, hardware, algorithm changes) – we are interested how they impact the results
- Random variables
- Uncontrolled variables

<table>
<thead>
<tr>
<th>Controlled variables</th>
<th>Experiment (Benchmark)</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random variables</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Uncontrolled variables</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The Challenge of Reasonable Repetition

- Variations
- Measurements must be repeated
- We want to eliminate the influence of random (non-deterministic) factors

Statistics

Controlled variables (e.g. compiler flags, hardware, algorithm changes) – we are interested how they impact the results

Random variables (e.g. hardware interrupts, OS scheduler) – we are interested in statistical properties of our results in face of these variables
The Challenge of Reasonable Repetition

- Variations
- Measurements must be repeated
- We want to eliminate the influence of random (non-deterministic) factors

Statistics

- **Controlled variables** (e.g. compiler flags, hardware, algorithm changes) – we are interested how they impact the results
- **Random variables** (e.g. hardware interrupts, OS scheduler) – we are interested in statistical properties of our results in face of these variables
- **Uncontrolled variables** – mostly fixed, but can cause bias of the results
Benchmark goal

- Estimate (a confidence interval for) the **mean** of execution time of a given benchmark on one or more platforms.
- The mean is the property of the probability distribution of the random execution times.
- We can only **estimate** the mean value from the measurements.
- Confidence interval is important:
  - CI of 95% ⇒ in 95% of cases, the true mean will be within the interval.
Measuring execution time

Levels of repetition

- Results variance occurs typically at multiple levels, e.g.:
  - (re)compilation
  - execution
  - iteration inside a program

- Sound benchmarking methodology should evaluate all the levels with random variations
Results variance occurs typically at multiple levels, e.g.:
- (re)compilation
- execution
- iteration inside a program

Sound benchmarking methodology should evaluate all the levels with random variations

How many times to repeat the experiment at each level?
- As little times as possible to not waste time
- As many times as possible to get reasonable confidence in results

How to summarize the results?
Significance testing

- Is it likely that two systems have different performance?
- This technique has significant problems, especially when used with results of computer benchmarks.
- ministat tool (FreeBSD)
Significance testing

- Is it likely that two systems have different performance?
- This technique has significant problems, especially when used with results of computer benchmarks.
- ministat tool (FreeBSD)
Measuring execution time

Summarizing benchmark results

- **Significance testing**
  - Is it likely that two systems have different performance?
  - This technique has significant problems, especially when used with results of computer benchmarks.
  - ministat tool (FreeBSD)

- **Visual tests**
  - Do the two confidence intervals overlap?

```
+------------------------------------------------------------+
| + |
| x ++ |
| x x ++ |
| xxxx ++ xx x ++ +|
| |________M_________A__________________| |
| |______M__________A_________________| |
| +------------------------------------------------------------+
```

↑ ministat shows standard deviation, not confidence intervals!

- No ⇒ different performance is likely
- Yes ⇒ more statistics needed
- Hard to estimate speedup and its confidence interval
Analysis of results should be statistically rigorous and in particular should quantify any variation. Report performance changes with effect size confidence intervals.
Repeating iterations

- We are interested in *steady state performance*

- Initialization phase
  - First few iterations typically include the initialization overheads
  - Warming up caches, teaching branch predictor, memory allocations

- Independent state
  - Ideally, measurements should be *independent, identically distributed* (i.i.d.)
  - Independent: measurement does not depend on any a previous measurement
  - Independent $\Rightarrow$ initialized
When a benchmark reaches independent state?

- Manual inspection of graphs from measured data
  - 1 run-sequence plot $\Rightarrow$ easy identification of initialization phase $\Rightarrow$ strip
When a benchmark reaches independent state?

- **Manual inspection of graphs from measured data**
  
  1. run-sequence plot ⇒ easy identification of initialization phase ⇒ strip

- **Independence assessment** – plot the following plots on original and randomly reordered sequence
  
  - lag plot (for several lags – e.g. 1–4)
  - auto-correlation function

- **Any visible pattern suggests the measurements are not independent**
Dependency of a measured values on the previously measured value.
Auto-correlation function

Left: dependent

Right: independent
Recommendations

Use this manual procedure just once to find how many iterations each benchmark, VM and platform combination requires to reach an independent state.

If a benchmark does not reach an independent state in a reasonable time, take the same iteration from each run.
What if different executions exhibit higher variance than iterations?

<table>
<thead>
<tr>
<th></th>
<th>bloaj6</th>
<th>eclipse9</th>
<th>lusearch9</th>
<th>tradebeans9</th>
<th>tradesoap9</th>
<th>xalan6</th>
<th>xalan9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration</td>
<td>14.1</td>
<td>0.8</td>
<td>3.3</td>
<td>1.5</td>
<td>0.8</td>
<td>7.0</td>
<td>3.5</td>
</tr>
<tr>
<td>Execution</td>
<td>3.7</td>
<td>0.4</td>
<td>30.3</td>
<td>0.4</td>
<td>0.4</td>
<td>9.1</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Determine initialized and independent state as before.
Measuring execution time » Repeating compilation

Repeating compilation

- Sometimes even a compiler can influence the benchmark results.
- Code layout generated by the compiler: original vs. randomized

Why code layout makes a difference?
- If you cannot control the factor, make it random.
Multi-level repetition

- We have to repeat the experiments to narrow confidence interval.
- If the variance occurs at higher levels (execution, compilation), we need to repeat at least at that level.
- Repeating at lower level may be cheaper (no execution overhead, compilation overhead, etc.)
  - Time can be saved by repeating at lower levels.
- How to find required number of repetitions at each level to reach given confidence interval?
- Can be formulated mathematically.
Notation

- **Levels**
  - Lowest level (iteration) = 1
  - Highest level (e.g. compilation) = $n$

- *Initial experiment*
  - bold letters
  - $r_1, c_1$

- *Real experiment*
  - normal letters
  - $r_1, c_1$
Initial experiment

Goal is to find the required number of iterations at each level.

- Select number of repetitions (exclusive of warm-up) $r_1, r_2, \ldots$ to be arbitrary but sufficient value, say 20.
- Gather the cost of repetition at each level (time added exclusively by that level, e.g. compile time)
  - $c_1$ iteration duration
  - $c_2$ time to gen an execution (time to independent state)
  - $c_3$ compilation time
- Measurement times: $Y_{j_1 \ldots j_1}, \quad j_1 = 1 \ldots r_1, j_2 = 1 \ldots r_2, \ldots$
- Calculate arithmetic means for different levels: $\bar{Y}_{j_n \ldots \ell}$
Variance estimators

- After initial experiments, $n$ unbiased variance estimators $T_1^2, \ldots, T_n^2$ is calculated.
- They describe how much each level contributes independently to variability in the results.
- Start with calculating $S_i^2$ – biased estimator of the variance at each level $i$, $1 \leq i \leq n$:

$$S_i^2 = \frac{1}{\prod_{k=i+1}^{n} r_k} \frac{1}{r_i - 1} \sum_{j_n=1}^{r_n} \cdots \sum_{j_i=1}^{r_i} (\bar{Y}_{j_n\ldots j_i\ldots} - \bar{Y}_{j_n\ldots j_{i+1}\ldots})^2$$

- Then obtain $T_i^2$:

$$T_1^2 = S_1^2$$

$$\forall i, 1 < i \leq n, T_i^2 = S_i^2 - \frac{S_{i-1}^2}{r_{i-1}}$$

- If $T_i^2 \leq 0$, this level induces little variation and repetitions can be skipped.
Optimum number of repetitions at different levels $r_1, \ldots, r_{n-1}$ can be calculated as:

$$r_i = \left\lceil \sqrt{c_i + 1} \frac{T_i^2}{c_i T_{i+1}^2} \right\rceil$$

Then recalculate: $S_n^2$ and $\bar{Y}_{j_n}, \ldots$ as before but with data from real experiment.

Asymptotic confidence interval with confidence $(1 - \alpha)$ is:

$$\bar{Y} \pm t_{1-\frac{\alpha}{2}, \nu} \sqrt{\frac{S_n^2}{r_n}}$$

where $t_{1-\frac{\alpha}{2}, \nu}$ is $(1 - \frac{\alpha}{2})$-quantile of the $t$-distribution with $\nu = r_n - 1$ degrees of freedom.
Real Experiment: Confidence Interval

- Optimum number of repetitions at different levels \( r_1, \ldots, r_{n-1} \) can be calculated as:

\[
\forall i, 1 \leq i < n, \quad r_i = \left\lceil \sqrt{\frac{c_i + 1}{c_i}} \frac{T_i^2}{T_{i+1}^2} \right\rceil
\]

- Then recalculate: \( S_n^2 \) and \( \bar{Y}_{jn} \) as before but with data from real experiment.

- Asymptotic confidence interval with confidence \((1 - \alpha)\) is:

\[
\bar{Y} \pm t_{1-\frac{\alpha}{2}, \nu} \sqrt{\frac{S_n^2}{r_n}}
\]

where \( t_{1-\frac{\alpha}{2}, \nu} \) is \((1 - \frac{\alpha}{2})\)-quantile of the t-distribution with \( \nu = r_n - 1 \) degrees of freedom.
Recommendation

For each benchmark/VM/platform, conduct a dimensioning experiment to establish the optimal repetition counts for each but the top level of the real experiment. Re-dimension only if the benchmark/VM/platform changes.
Measuring execution time
- Repeating iterations
- Repeating executions
- Repeating compilation
- Multi-level repetition

Measuring speedup
Measuring speedup

- Speedup is a ratio of two execution times (random variables)
- What is the speedup confidence interval?
- How many times to repeat the speedup experiments?
Speedup confidence interval

- $\bar{Y}$ – old system execution time
- $\bar{Y}'$ – new system execution time
- Speedup: $\bar{Y}'/\bar{Y}$

\[
\bar{Y} \cdot \bar{Y}' \pm \sqrt{(\bar{Y} \cdot \bar{Y}')^2 - (\bar{Y}^2 - h^2)(\bar{Y}'^2 - h'^2)} / \bar{Y}^2 - h^2
\]

\[
h = \sqrt{t^2_{\alpha/2, \nu} \frac{S^2_n}{r_n}} \quad h' = \sqrt{t^2_{\alpha/2, \nu} \frac{S'^2_n}{r_n}}
\]
Relation of confidence interval of the speedup to confidence interval on individual measurements:

\[ e \approx \frac{\bar{Y}'}{\bar{Y}} \sqrt{e^2 + e'^2} \]

- \( e, e' \) half-width of the old resp. new confidence interval
Always provide effect size confidence intervals for results. Either for single systems or for speedups.