Performance Evaluation

Francieli Zanon Boito

Motivation

- So far: we talked about performance (a lot)
  - But how do we measure it?
- Today: discuss how to conduct a performance evaluation
  - Why? So we can trust our results and get to correct conclusions
  - Useful conclusions that can be reproduced by others
  - These conclusions may guide important decisions
  - Avoid wasting time and money!
  - Our intuition is not always accurate
- Next lab session: performance evaluation!

Step 1: planning an experiment

- Start by formulating the hypotheses
    - Design experiments accordingly
- Example: if our hypothesis is "a heap is the best data structure to implement algorithm A"
    - The experiment goal is to "evaluate different data structures to implement algorithm A"
    - And NOT to "prove the heap is the best"
- Sometimes we don't know exactly what we are looking for, and that is okay in some cases
    - Draw some hypotheses, design new experiments to test them

Planning the experiment

- What are the variables that affect performance in this situation?
  - Number of tasks/processes/threads,
  - Size of tasks/messages/blocks,
  - Network topology,
  - Version of libraries/compilers,
  - Etc.
- Some variables are harder to control
  - For instance: interference from other users
- What do we want to compare?
- What metrics do we need to collect?
- **No amount of statistical analysis can fix a bad experiment**

ALWAYS take notes!

- Keep some kind of lab book
- Document your line of thought
  - Why you are doing what you are doing
- So you:
  - don't forget
  - can eventually share it with others
  - are sure it makes sense
  - don't have a hard time to write a paper/prepare a presentation/talk with your advisor
- Find a tool and a frequency that works for you
  - For instance: while waiting for something to compile/run, take the time to write

Not only for ideas

- Take notes of everything that could affect your experiment
  - All software versions, on what nodes it executed, in what order, at what times, …
  - If pseudo-random number generation is involved, store the seeds
- Detailed instructions to allow reproducible results
  - Also to explain why results can't be reproduced (or are not what we expected)

Avoid handicraft

- DON'T manipulate data/run experiments by hand
  - Humans are error-prone
- Write scripts for everything
  - Less likely to make mistakes
  - Or at least you (or others) can later find your mistakes

Take care of your code

- All your code (including scripts) has to be under version control
    - Minor changes can have major impacts
    - Can't reproduce results if we don't have the codes that generated them
- Don't forget to backup!
- Typical option: bitbucket, github, gitlab, etc

Handle variability

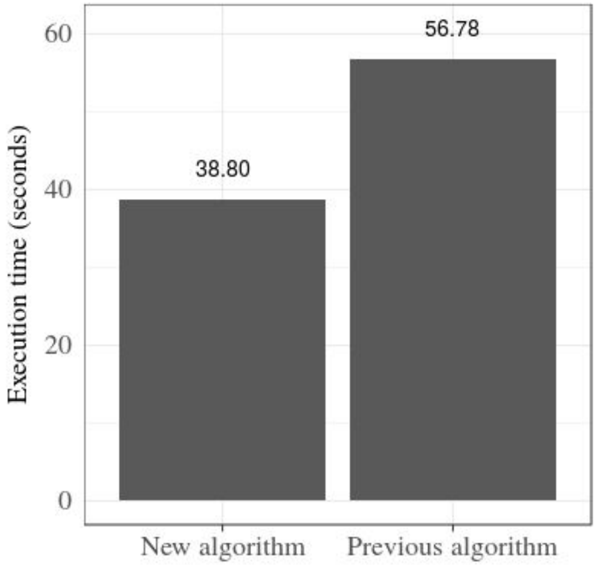
- The same application, on the same machine: not always the same execution time
  - DVFS, cache, compiler optimizations, etc
- We need some replications
  - But simply repeating the test in a for loop could induce bias
  - Populate caches, increase the processor frequency, other activities in the system, etc
  - Try different configurations, randomize, run at different times and average-away or analyze separately
  - Afterwards: check the sequence of replications

Step 2: Analyze results

- "So I did everything right so far, are my conclusions sound?"
    - You actually have a higher probability of reaching good conclusions
    - As long as you analyze your results correctly

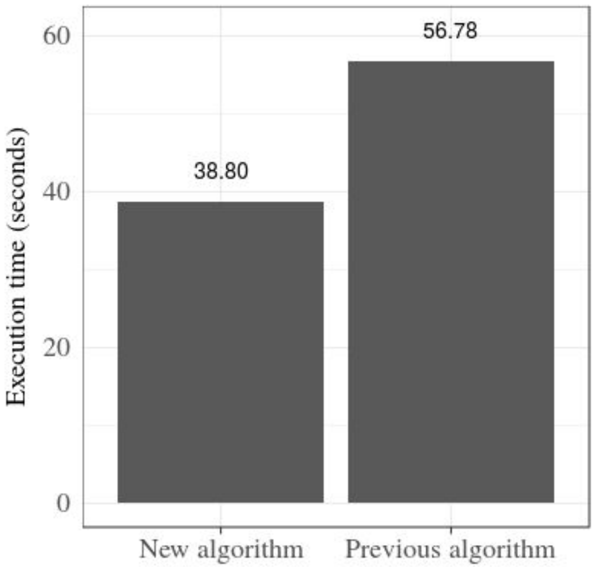
Descriptive analysis

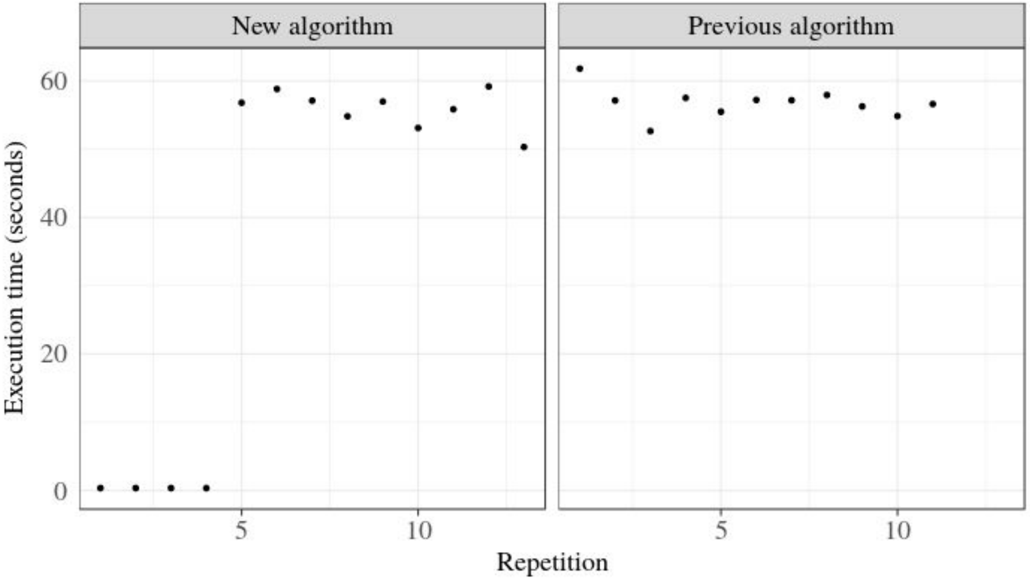
- First: get to know your data
  - Plot it
  - Look for tendencies or anomalies



Descriptive analysis

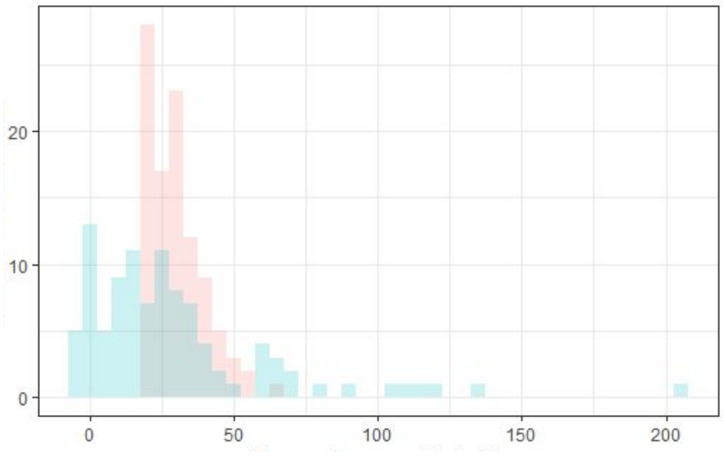
- First: get to know your data
  - Plot it
  - Look for tendencies or anomalies

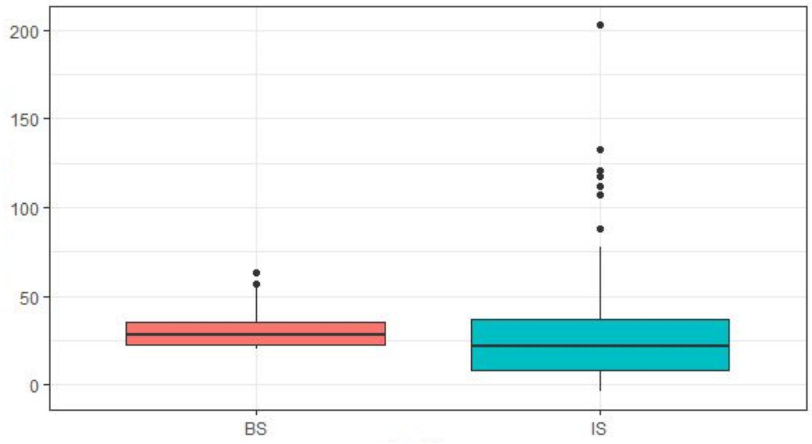




Descriptive analysis

- First: get to know your data
  - Plot it
  - Look for tendencies or anomalies
  - Check the distribution





Metrics

- The mean is 30 seconds for both

Alg A	Alg B
10	30
12	30
13	30
15	30
100	30

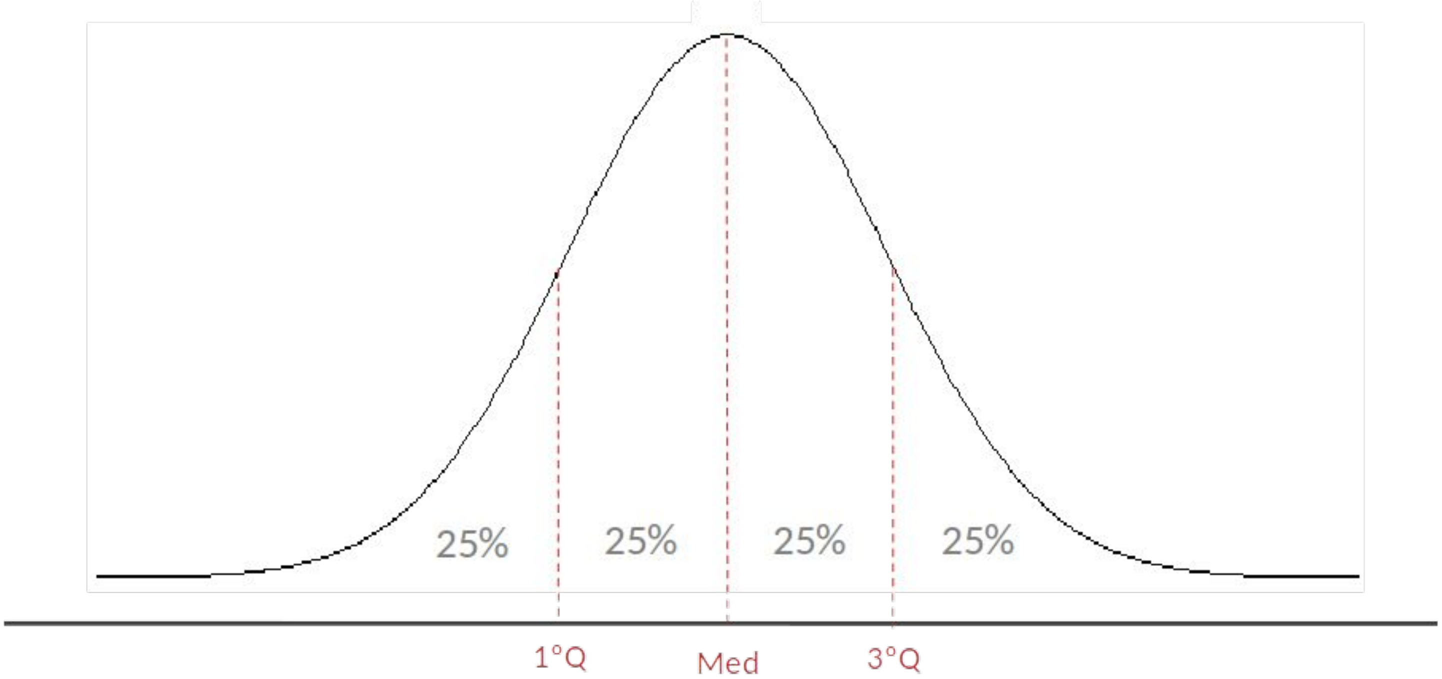
Metrics

- The mean is 30 seconds for both
- The median for A is 13, and for B is 30 (50% of the execution times for A are of up to 13)

Alg A	**Alg B**
10	30
12	30
13	30
15	30
100	30

Metrics

- We could also be interested in the mode
- Or the first and third quartiles



Metrics

- We could also be interested in the mode
- Or the first and third quartiles
- It is useless to report mean/median/mode without any information about the variability

Keep all your data

- "Heuristics" to avoid (unless they are justified):
  - Discard the N best and worst results
  - Take only the best N measurements
- DON'T just calculate the mean/median and discard everything

Statistical inference methods

- Our measurements are samples
- One average is higher than the other… it does not necessarily mean anything
- Statistical inference: draw conclusions about the population from samples
    - Error bars
    - Hypothesis testing (example: test if the distribution is normal)
    - Need to define a significance before testing
    - Parametric vs. non-parametric methods
    - We always assume observations are independent (need to ensure that)

Present your results

- Choose the right type of graph
  - Does connecting dots with lines make sense?
- Be clear
  - Useful information in the graph legend and axes
  - All the details about how those numbers were obtained (this is also part of your code)
- Be wise (and honest) about the axis scales
  - If unexpected scale, make it clear
  - Graphs that are presented together have the same scales (otherwise explain it)

Further reading

- Materials by Arnaud Legrand for the "Scientific Methodology and Performance Evaluation" course: https://github.com/alegrand/SMPE/tree/master/sessions/2018_10_Grenoble/
- Spurious correlations: http://tylervigen.com/spurious-correlations
- Books:
  - Raj Jain, "Art of Computer Systems Performance Analysis"
  - Wayne Booth et al., "The Craft of Research"