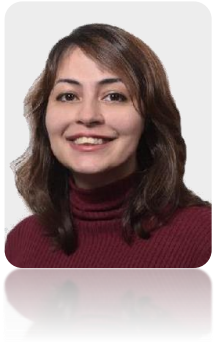
The Art of Measurement†

IN4390 Quantitative Evaluation of Embedded Systems

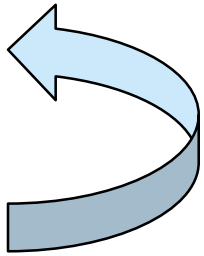
Koen Langendoen



†Original slides by Mitra Nasri, now at TU/e

Course outline

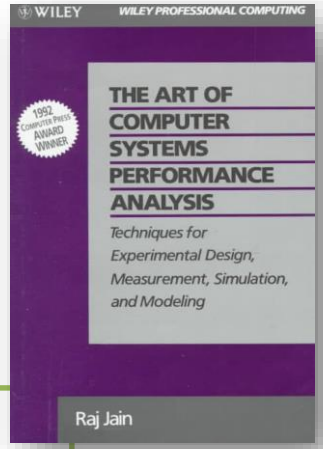
- Lect 1: terminology, non-functional properties
- Lect 2+3: Design of Experiments
- Lect 4: The Art of Measurements



Agenda

- **The art of measurement**
  1. Pillars of measurement-based evaluation
  2. Goals and metrics (book chapter 3)
  3. Workload selection/generation (book chapter 4)
  4. Measurement: practical considerations
  5. The art of data presentation (book chapter 10, 11)
  6. Confidence level and confidence interval (book chapter 13)
  7. Summarizing measured data (book chapter 12)

- **Self study**
  - Common mistakes (book chapter 1, 2)
  - Summarizing measured data (book chapter 12)
  - The art of data presentation (book chapter 10, 11)



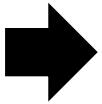
Source:
- Raj Jain, "The Art of Computer Systems Performance Analysis"

An overview of evaluation techniques

- **Analytical modeling**
  - Based on a rigorous mathematical models
  - Lectures 7, 9-13 (MC + QT)

- **Simulation**
  - Simulate the system operation
    (usually only small parts thereof)
  - Not covered in this course

- **Measurement**
  - Implement the system in full and measure its performance directly
  - This lecture

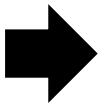


- **Positive**: Provides the best insight into the effects of different parameters and their interaction
  - example: is it better to configure the system with one fast disk or with two slow disks?
- **Positive**: Can be done before the system is built and takes a short time
- **Negative**: these models are rarely accurate
  - Usually needs many simplifying assumptions
  - Depends on the quality and correctness of these assumptions

read more here: https://tu-dresden.de/zih/ressourcen/dateien/lehre/ws1112/lars/vorlesungen/lars_lecture_02_requirements-metrics-techniques.pdf?lang=en

An overview of evaluation techniques

- **Analytical modeling**
  - Based on a rigorous mathematical models
  - Lectures 7, 9-13 (MC + QT)
- **Simulation**
  - Simulate the system operation (usually only small parts thereof)
  - Not covered in this course
- **Measurement**
  - Implement the system in full and measure its performance directly
  - This lecture

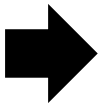


- **Positive**: full control of simulation model, parameters, level of detail (high flexibility).
- **Positive**: Can be done before the system is built
  - Simulation of a full system is infeasible
  - Simulation of the system parts does not take everything into account
- **Negative**:
  - It may still include simplifying assumptions

read more here: https://tu-dresden.de/zih/ressourcen/dateien/lehre/ws1112/lars/vorlesungen/lars_lecture_02_requirements-metrics-techniques.pdf?lang=en

An overview of evaluation techniques

- **Analytical modeling**
  - Based on a rigorous mathematical models
  - Lectures 7, 9-13 (MC + QT)
- **Simulation**
  - Simulate the system operation (usually only small parts thereof)
  - Not covered in this course
- **Measurement**
  - Implement the system in full and measure its performance directly
  - This lecture



- **Positive**: The most convincing
  - Numbers don't lie (but statistics do ☺)
- **Negative**: has high costs
  - Requires a full implementation of the system (buying hardware)
  - What if you want to change the hardware in case the performance is bad?

read more here: https://tu-dresden.de/zih/ressourcen/dateien/lehre/ws1112/lars/vorlesungen/lars_lecture_02_requirements-metrics-techniques.pdf?lang=en

Pillars of Measurement-Based Evaluation

Pillars of measurement-based evaluation

What is the goal?

Can you achieve what you want to achieve by measurement?

What to measure?

- Metrics
- Workload

Does your metric really measure what you want to measure?

What type of input workload should you consider?

How to measure?

- DOs and DON'Ts of measurements

What tools should/can you use for measuring a metrics?

Considerations while doing measurement

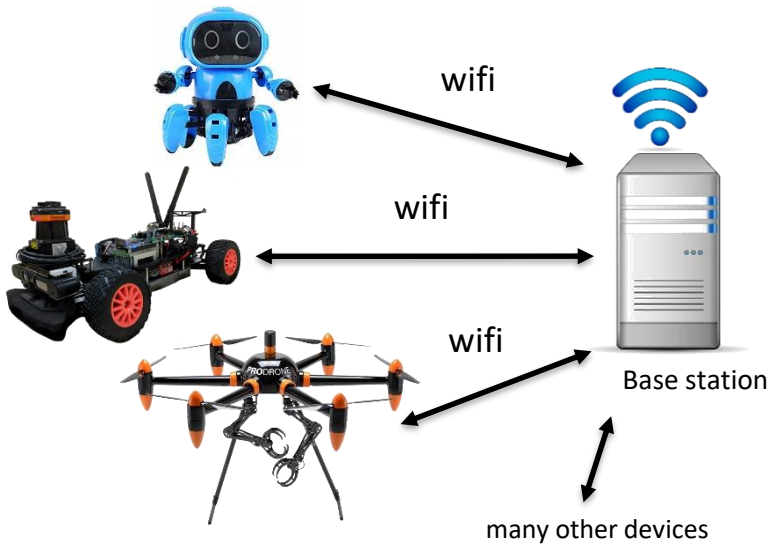
How to summarize and present the output?

How to summarize your thousands of data points?

What are the statistical methods/concepts that can help you?

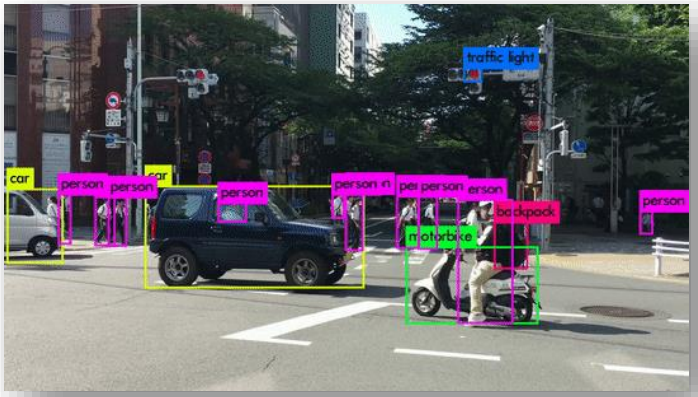
Goal

What do you want to evaluate?



Object-detection service

- Input: image
- Output: position and type of the object



Goal: no human will ever be harmed by my system

Question: What should you measure to ensure this property?

Question: What type of non-functional property is this?

Assume that your object-detection algorithm is 100% accurate and never does any mistake!

Metrics

Metric types

Examples?

- a count of how many times an event occurs
- the duration of some time interval
- the size of some parameter

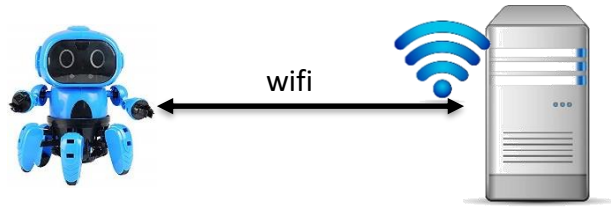
Metrics can have conditions

- Number of successfully processed images in one hour
- Total execution time of the image processing task for serving 100 images
- Time until the next failure

Metric and goal: example

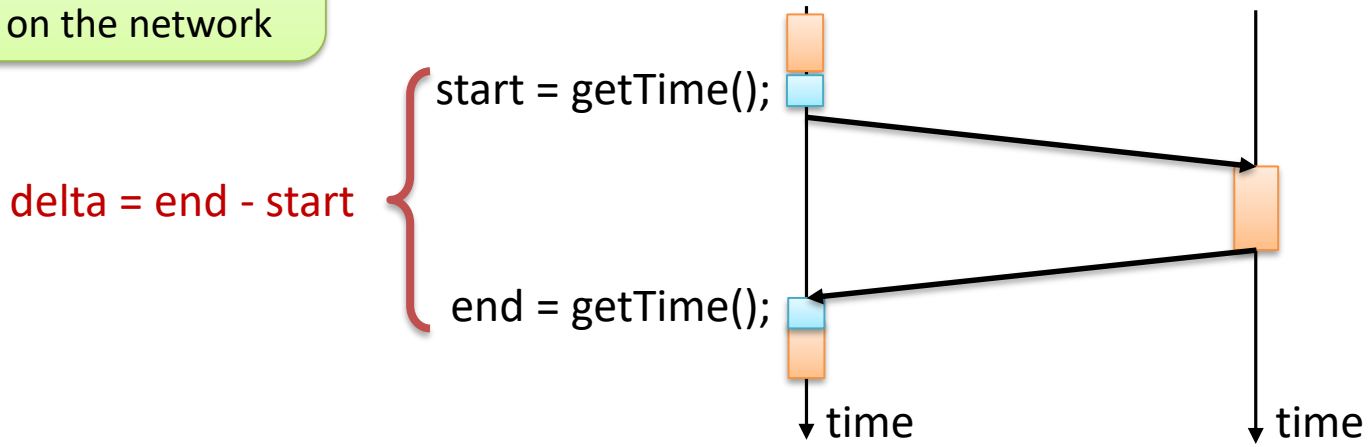
Question: What is being measured?

Answer: the round-trip time = response time of the object detection service plus the transmission time on the network

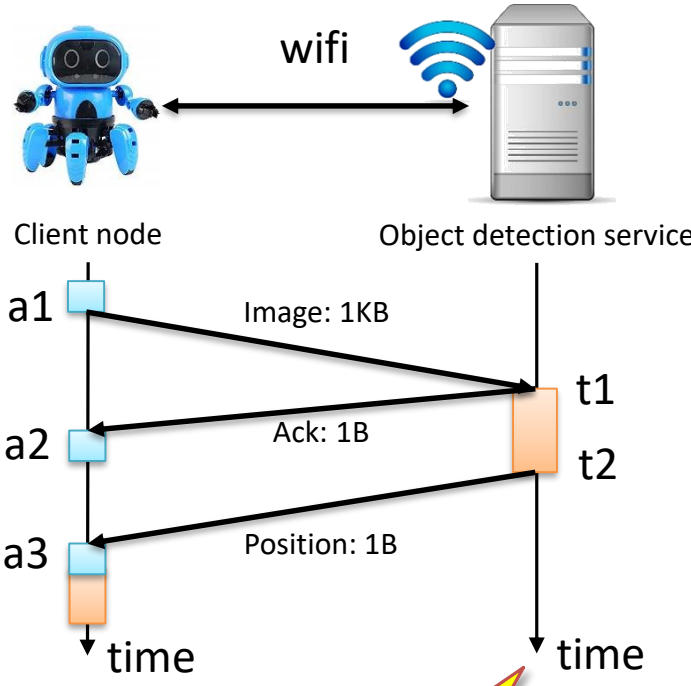


Client node

Object detection service



Metric and goal: example



Question: is "B" the execution time of the object detection?

Answer: No.
B includes two network delay values. The network delay varies a lot (hence, the delay suffered when sending the Ack might be different from the one for sending the position).

Question: is "t2-t1" the execution time of the object detection?

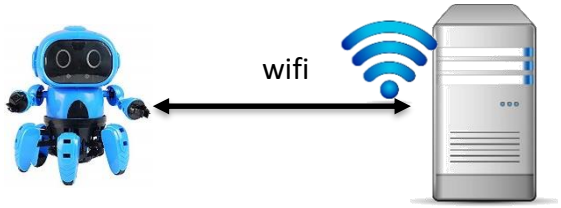
Answer: No.
t2-t1 is the response time of the object detection method and it includes other delays such as preemptions by other higher-priority tasks on the server

System parameters (configurations)

- System parameters are configurations and parameters that impact a system's performance independent of the system's input workload

Example:

Metric is the round trip time of the object tracking



Sends its position to the base station

Navigates the device through the campus

Question: what are the system parameters in our example?

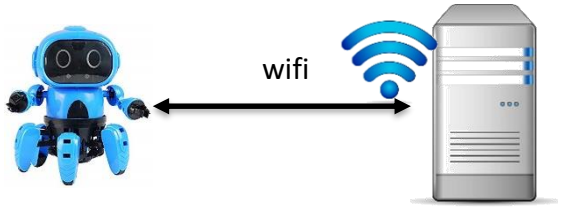
System parameters for our example:

- Speed of: base station's CPU, robot's microcontroller, and network.
- Scheduling policy on both ends.
- Network protocol.
- Operating system overhead for interfacing with the network.
- Reliability of the network affecting the number of retransmissions required.
- …

Workload model

- Workload model defines the model of input workload to the system
- It is usually a representation of the actual usage of the system after deployment

Question: what are the workload parameters?



Workload parameters for this example:

- Number of IoT devices (user)
- Frequency of sending requests to the base station
  - and distribution (periodic, Poisson, …)
- Complexity of the navigation task at different locations
  - Example: around cross roads is harder than straight roads
  - Proximity (presence) of other devices when it reaches to a cross road
- Other load on the robot's microcontroller and base station CPUs
- Other load on the network

Factors

- **Factors**: Parameters that are varied during the evaluation
- **Levels**: Values of a factor

Example:

- Image size: {10KB, 100KB, 1MB}
- Frequency of calls to service:
  - Periodic: 1 every {1, 10, 100, 1000} milliseconds
  - Exponential distribution with rate: {0.1, 0.2, 0.3, 0.4, 0.5}

- Service: image rotation
- Input: image and angle
- Output: rotated image

Assumption:

- Fixed: type of CPU and operating system.
- Measure under no other load on the hosts and the network.

Measuring Time: Practical Considerations

Image source: https://phys.org/news/2017-03-scientists-animals-year.html

Measuring time

Don’t measure by hand!

Measuring by hand: “looking" at a clock, launching the program, “looking" at the clock again when the program terminates

Approaches

1. **Measuring a process’s execution time**
   - UNIX’s “time” command
2. **Measuring time inside a program**
   - Programming language / OS support
3. **Measuring time from outside of the system**
   - @inside: when the event (of interest) happens, trigger an I/O pin.
   - @outside: monitor the I/O pin for changes to record the event time

UNIX time command

You can put time in front of any UNIX command

Invoked command

```
surf:~$  /usr/bin/X11/time  ls  -la  -R  ~/ > /dev/null

2.11user  1.94system  1:8.72elapsed  2%CPU
(0avgtext+0avgdata  0maxresident)k
0inputs+0outputs  (0major+1155minor)pagefaults  0swaps
```

- 2.11 seconds of **user time**
- 1.94 seconds of **system time**
- 1 minutes and 8.72 seconds of **wall-clock time**
- 2% of CPU was used

- 0+0k memory used (text + data)
- 0 input, 0 output (file system I/O)
- 1155 minor pagefaults
- 0 swaps

- **User Time**
  - time spent executing user code
- **System Time**
  - time spent executing kernel code (inside OS)
- **Wall-Clock Time**
  - time from start to end

See more here: https://en.wikipedia.org/wiki/Time_(Unix)

UNIX time command

```
surf:~$  /usr/bin/X11/time  ls  -la  -R  ~/ > /dev/null

2.11user  1.94system  1:8.72elapsed  2%CPU
(0avgtext+0avgdata  0maxresident)k
0inputs+0outputs  (0major+1155minor)pagefaults  0swaps
```

- **User Time**
  - time spent executing user code
- **System Time**
  - time spent executing kernel code (inside OS)
- **Wall-Clock Time**
  - time from start to end

Question: Why wall-clock ≥ User + System?

1. because the process can be preempted by other processes that run concurrently on the system
2. because the process can be blocked waiting for I/O

System time

Question: When do you get a large "system time"?

Larger system time may be a sign of higher calls to system APIs such as fopen(), fread(), fwrite(), etc.

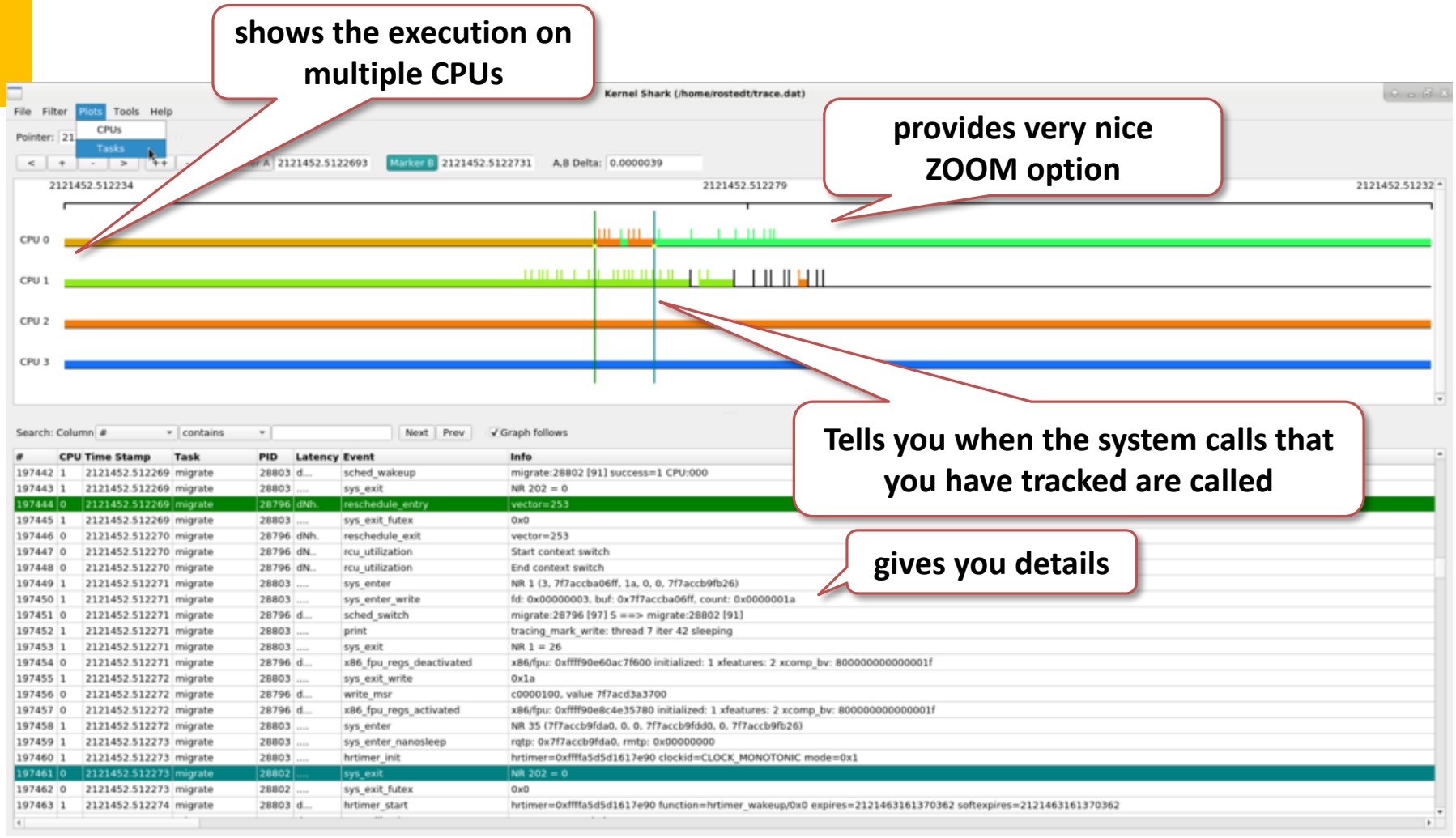
To pinpoint which system calls are more often used, you need to track them via separate tracing tools such as **ftrace**

KernelShark

- **A visualization tool to track "ftraces"**
- **Docs:** https://kernelshark.org/Documentation.html
- **Git:** https://git.kernel.org/pub/scm/utils/trace-cmd/trace-cmd.git/

More information: https://www.kernel.org/doc/Documentation/trace/ftrace.txt

Visualizing ftrace outputs using KernelShark



Wall-clock time

Question: What else does impact the wall-clock time other than system and user times?

Wall clock – system – user =
I/O + preemption + suspension + other interferences

- **I/O**: time waiting for performing an I/O interaction (including reading from or writing to files, ports, etc.)
- **Preemption**: time waiting for other tasks to execute
- **Suspension:** time being blocked (e.g., when a task wants to access a data that is protected by a lock) or being suspended due to an I/O operation
- **Other interferences:** time spent on accessing memory bus of a memory bank, etc.

Question: How can you control the impact of these delays on your measurements?

Wall-clock time

Wall clock – system – user =
I/O + preemption + suspension + other interferences

One approach is to try to run your process in **isolation** (when no other process is running)

Question: Would this be enough?

hey, I have designed a stabilizer that processes 1000 samples per millisecond on an Arduino Mega

Oh, even when your drone is flying?

No way! It happens only in the lab when I turn off every other process in the system!

Wall-clock time

```
surf:~$    /usr/bin/X11/time    ./parallelQuicksort2

9.76user  10.51system  0:06.11elapsed  331%CPU
(0avgtext+0avgdata 158268maxresident)k
0inputs+0outputs (0major+7599minor)pagefaults  0swaps
```

Question: Why wall clock time < system + user time?

This system could have multiple cores and execute the program in parallel on multiple cores. In that case, the wall-clock time becomes smaller than user or system times.

Drawbacks of UNIX time command

- **The time command has poor resolution**
  - “Only" milliseconds
- **“time” times the whole code**
  - Often ES code is not terminating (i.e. runs forever)
  - Sometimes we're only interested in timing some part of the code, for instance the one that we are trying to optimize

Timing with gettimeofday

- gettimeofday from the standard C library
- Measures the number of microseconds since midnight, Jan 1st 1970, expressed in seconds and microseconds

Can be used to time sections of code

- Call gettimeofday at beginning of section
- Call gettimeofday at end of section
- Compute the time elapsed in microseconds:
  (end.tv sec*1000000.0 + end.tv usec -
  start.tv sec*1000000.0 - start.tv usec) / 1000000.0)

Have a look at clock_gettime()

Measures the number of nanoseconds since midnight, Jan 1st 1970, expressed in seconds and nanoseconds. Obviously not precise at the nanosecond level but much better than gettimeofday

https://linux.die.net/man/2/gettimeofday

https://linux.die.net/man/3/clock_gettime

Using timers

Assumption: There is no other process or an operating system in this example. The goal is to measure the runtime of the middle instruction that runs on an Arduino Mega.

```
long x = 30, start, end;
start = micros();
x = 42 * x;
end = micros();
printf(end - start);
```

micros() returns values that are integer multiples of 4 microseconds

What can invalidate this measurement?

Each instruction is about 2~3 microsecond

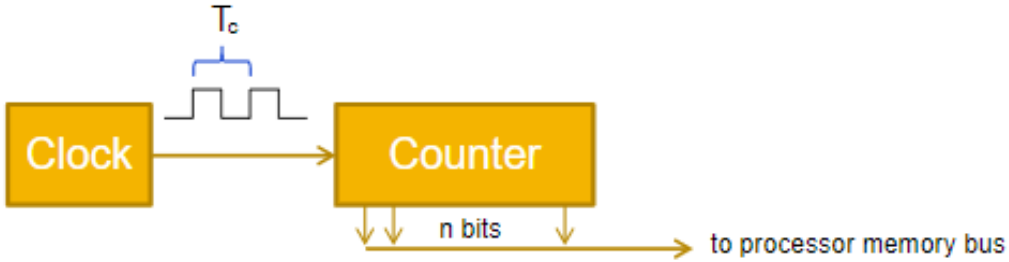
- **Timers are not infinitely accurate**
  - All clocks have a granularity
  - The error in a time measurement, even if everything is perfect, may be the size of this granularity (sometimes called a clock tick)

- Always know your clock granularity
- Ensure that your measurement is for a long enough duration (say 100x the "tick")

source: http://wgropp.cs.illinois.edu/courses/cs598-s16/lectures/lecture06.pdf

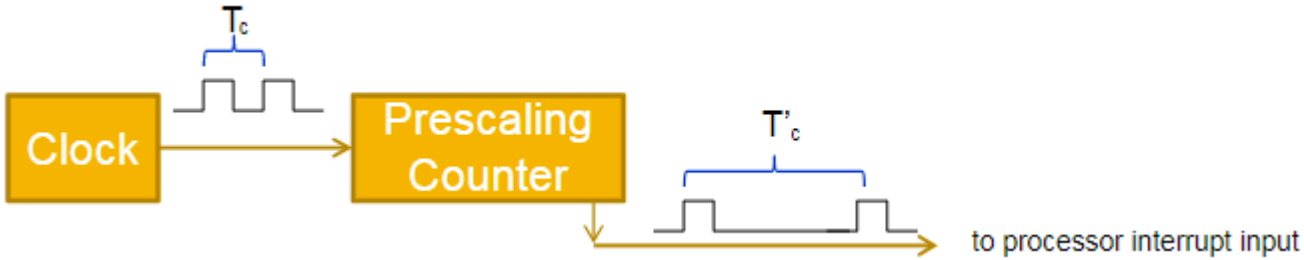
Timers and rollover

Hardware timer



Counter value is read from a memory location.

Software timer



When interrupt occurs, interrupt-service routine increments the timer value which is read by a program.

Timer Rollover

- Occurs when an n-bit counter undergoes a transition from its maximum value $2^n - 1$ to zero.

Timers and rollover

- high resolution
- early rollover

- low resolution
- late rollover

T'_c	32-bits	64-bits
10 ns	42 s	5058 years
1 μs	1.2 hour	0.5 million years
1 ms	49 days	0.5×10^9

There is a trade-off between rollover time and granularity of the timer

The Art of Data Gathering

Image source: https://streetfightmag.com/2019/03/25/the-fundamental-ethical-stakes-of-data-collection-and-ad-targeting/#.Xc68AvlKhaQ

Why should you repeat your measurements?

Errors

Error is the difference between the **measured value** and the 'true value' of the thing being measured

Uncertainties

Uncertainty is a quantification of the *doubt about the measurement result*

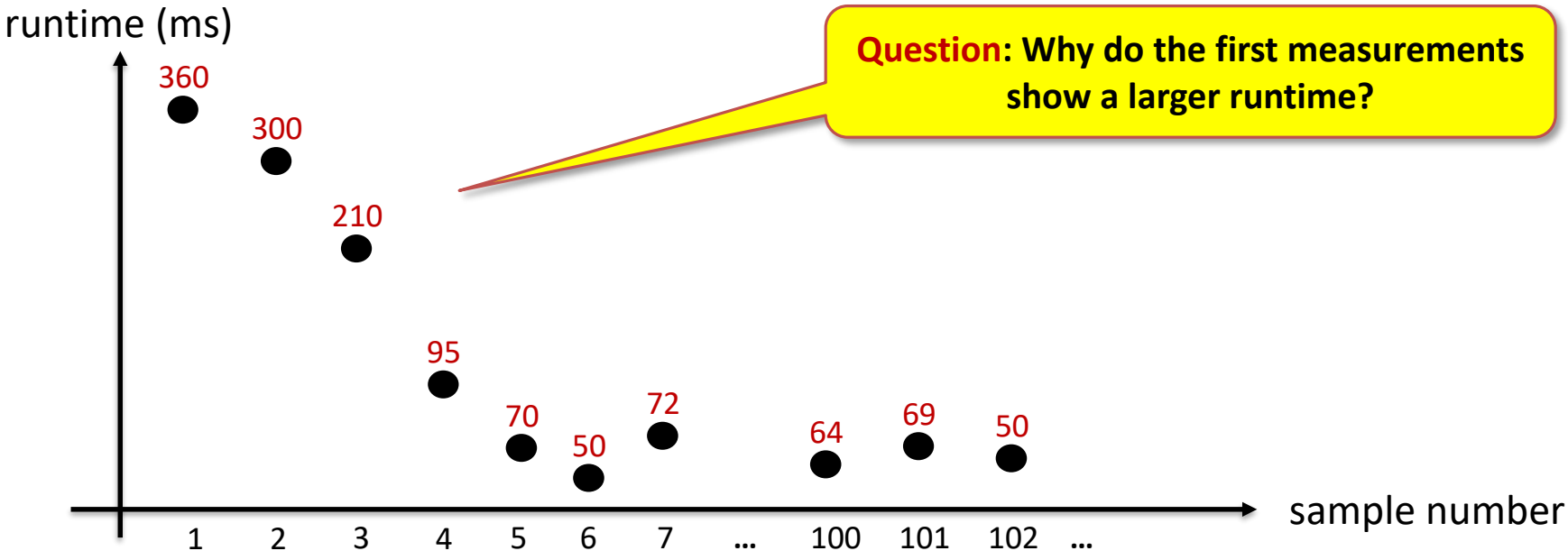
Error can be a caused by a low timer resolution

Network delay and signal strength are two sources of **uncertainty** in this example

- Whenever possible, we try to correct for any known errors
- But any error whose value we do not know is a source of uncertainty

Measurement process

Assumption: there is no other process in the system. There is no operating system.
The measurements are the runtime of the only task in the system.



The reason is “cold start”

Cold start

- Code may still be on disk, and not even loaded into memory.
- Data may be in slow memory rather than fast, e.g., cache, (note: this may be wrong or right for what you are measuring)

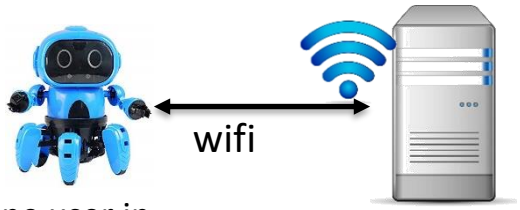
Question: How to solve this issue if we want to measure the long-run average?

Let the program run for a while and then start measuring

Note: usually you need to put effort to ensure that the data in the intended part of the memory hierarchy.

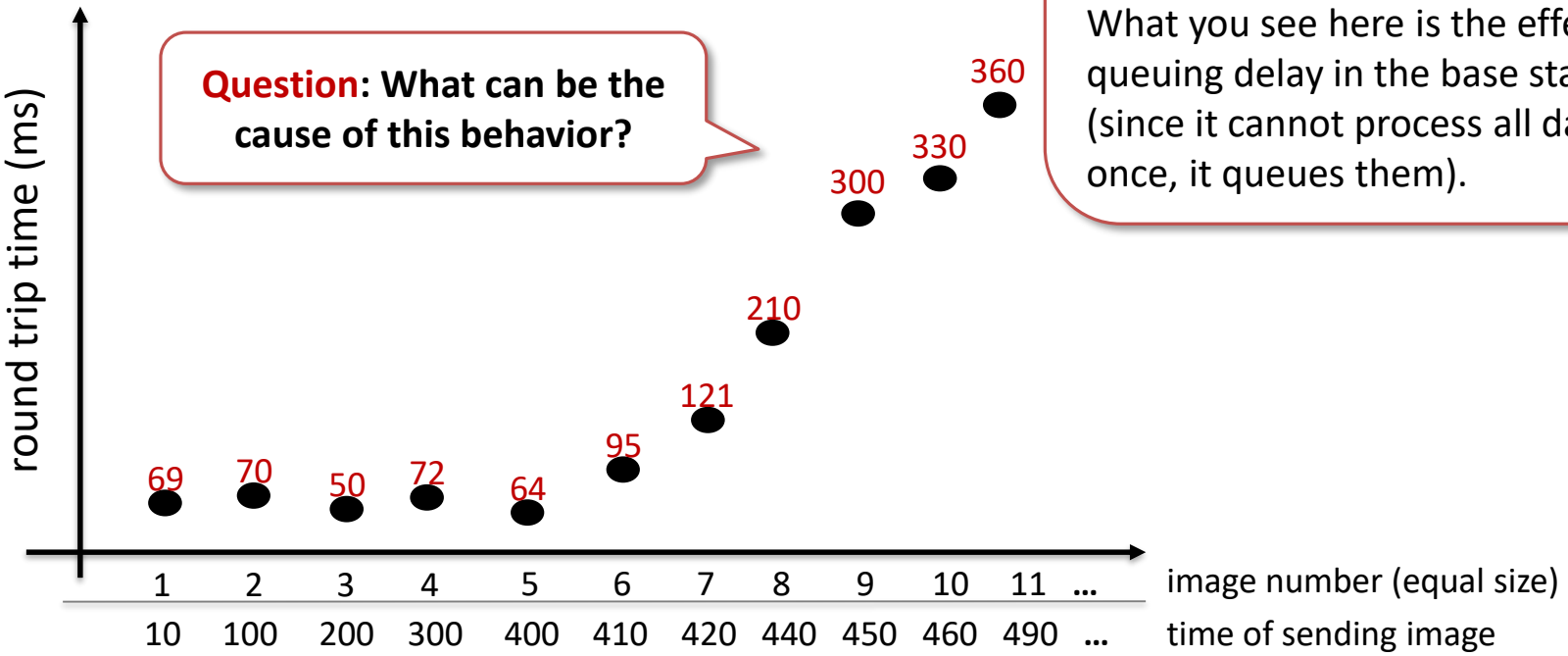
Considerations for gathering data

- **The effect of hidden or ignored parameters**



one user in the network

Question: What can be the cause of this behavior?



Note: In this example, measured values are affected by the frequency of sending images.

What you see here is the effect of queuing delay in the base station (since it cannot process all data at once, it queues them).

Considerations for gathering data

Question: What is it?

- **Measurement-induced perturbations**
  - The system resources consumed by the measurement tool itself as it collects data may strongly affect the system's performance.
  - Tracing produces the highest level of perturbation (both CPU and disk are used) on
    - time measurements,
    - spatial and temporal memory access (cache flush, different paging,. . .), or
    - system response time (and thus on workload characterization).
  - The larger the overhead, the more likely the system behavior will be modified.

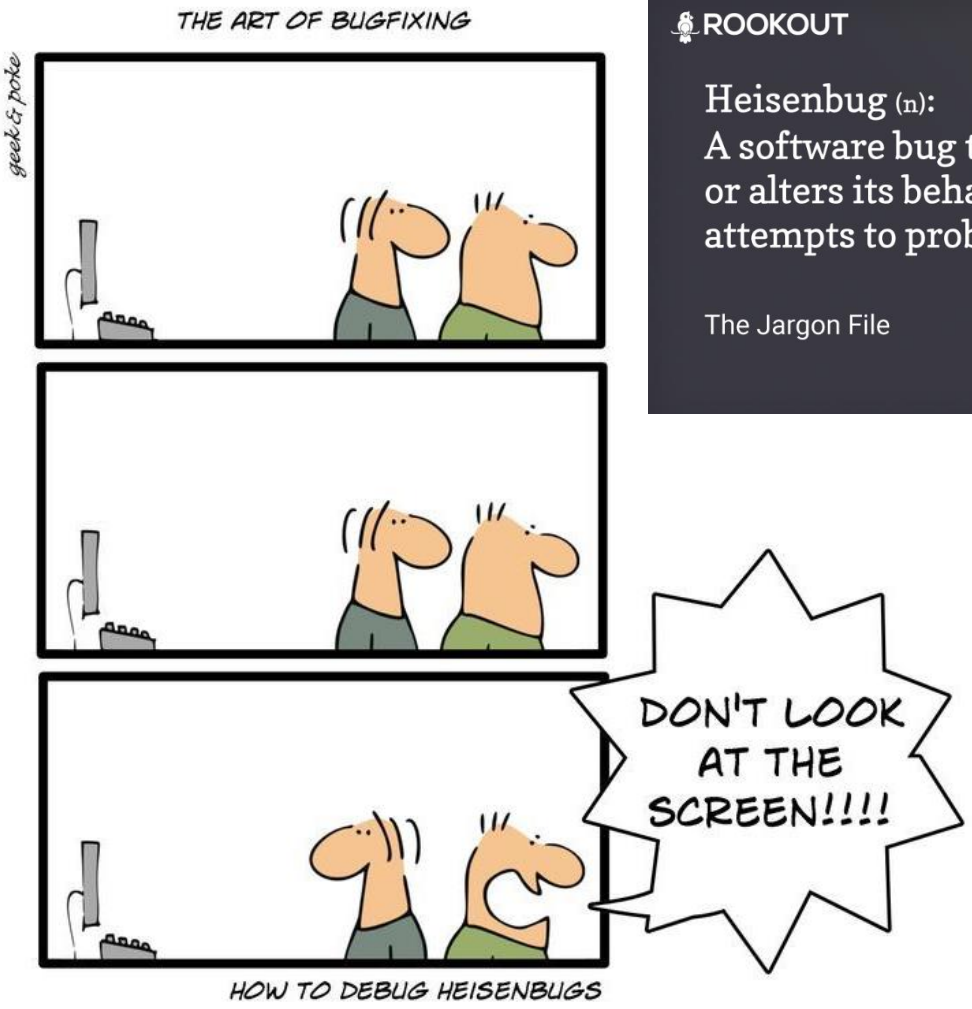
On embedded systems, writing outputs on the terminal will heavily impact the system's behavior

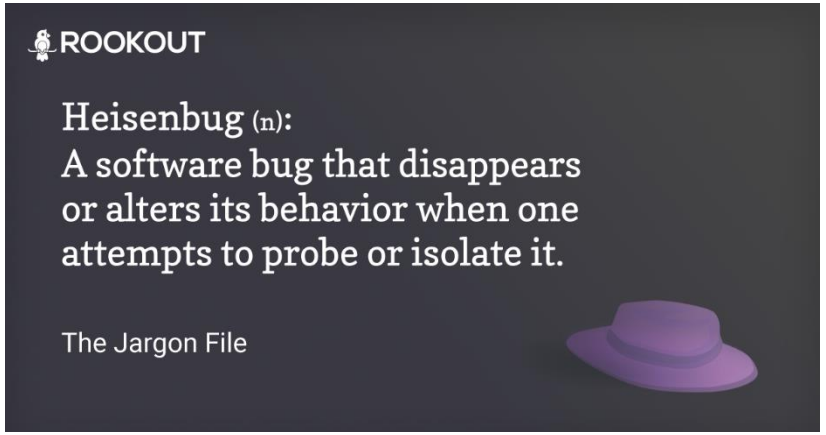
In Arduino Mega, storing time in an array takes about 2~3 microseconds, while writing it using printf() is about 70~80 microseconds

Considerations for gathering data

Question: What is it?

- **Measurement-induced perturbations**





Considerations for gathering data

Question: how to solve it?

When measuring time:

- **Do not write the measured data on the output while gathering samples**
  - **Solution:** store the data in an array and then output the array after gathering samples

- **Do not use dynamic memory allocation for storing samples (e.g., malloc())**

Question: Why?

- It messes up the memory alignment of the program and changes its behavior.
- You may even run into "out of memory" problem due to fragmentation of memory

Solution: Use an array with predefined size (static memory allocation)

Problem: smart compiler

If the result of the computation is not used, the compiler may eliminate the code.

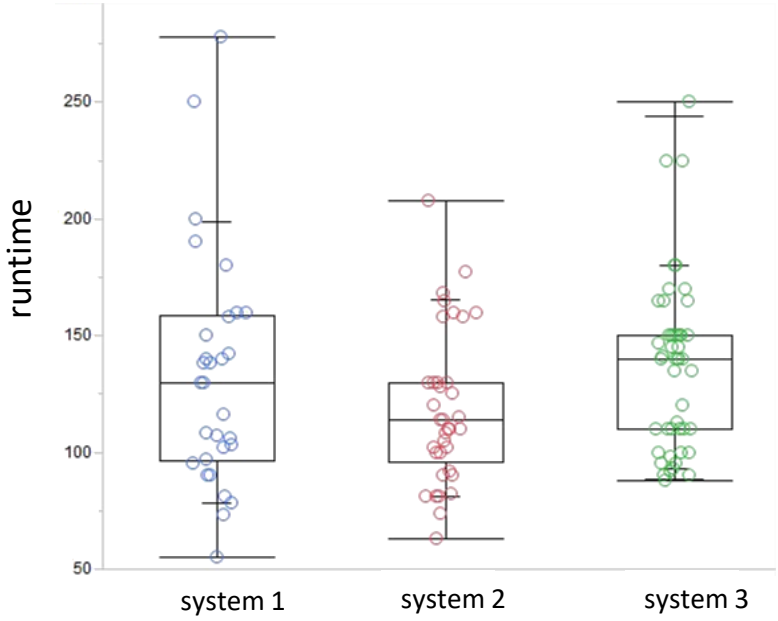
```
void bar(){
long start = getTime();
…
x = (y > 1) ? (y * y * 2) : (2.123456789 % 17);
…
// foo(x);
….
long end = getTime();
print(end – start);
}
```

Question: When does it become an issue?

The compiler may remove the line that changes x, so your final measurement may look much shorter than it will be if “foo” was not commented out

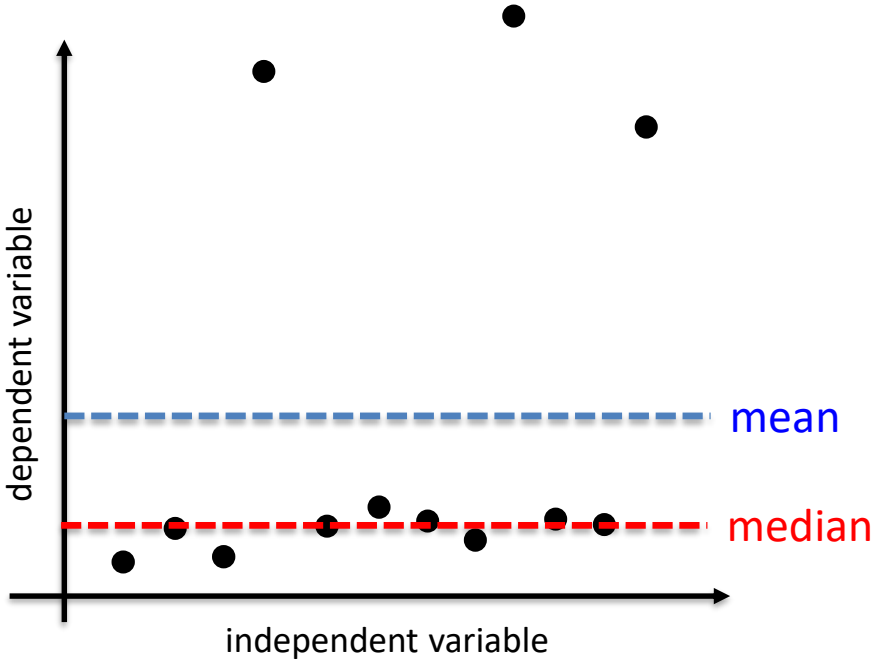
You decided to comment out this line to avoid including the runtime of “foo” into “bar”.

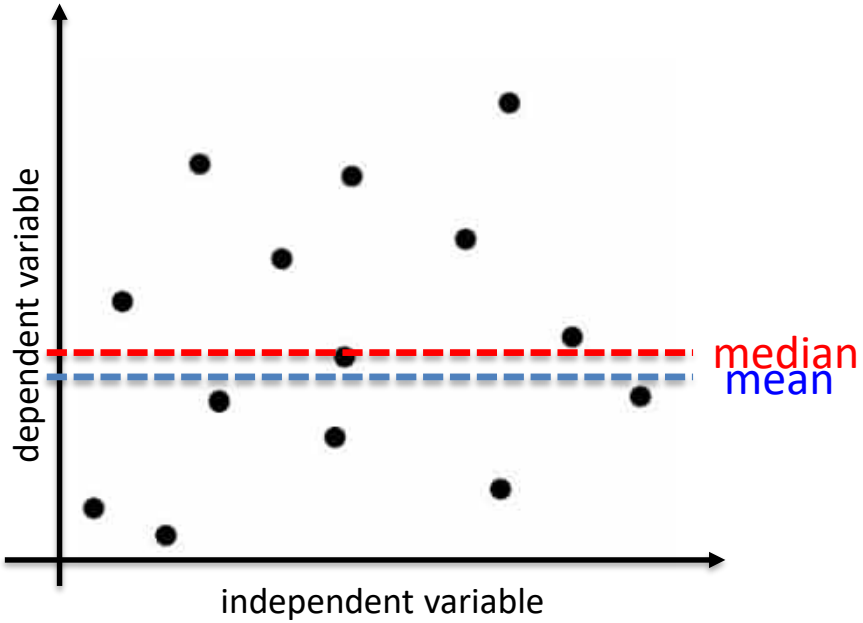
The Art of Data Summarization



Data summarization

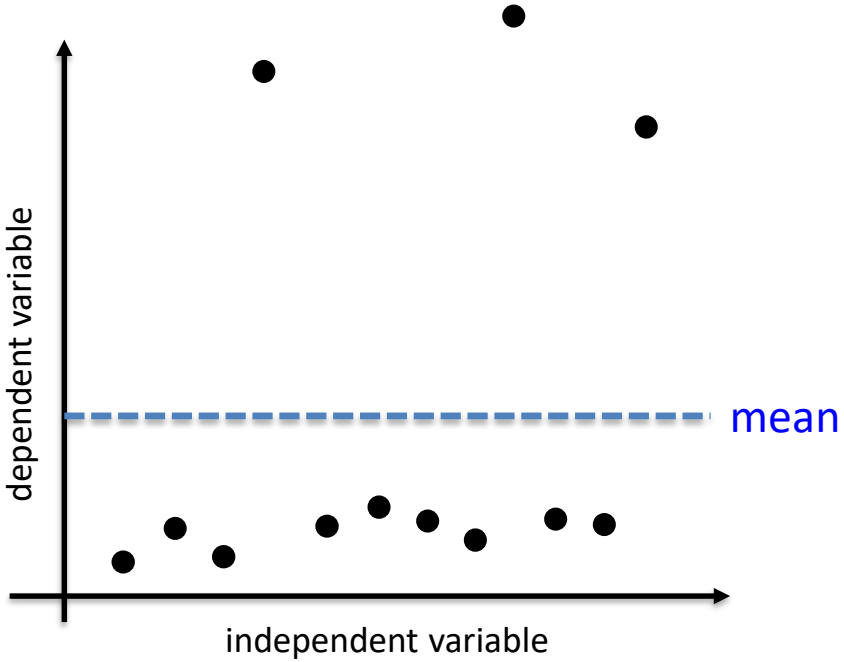
1. How to report the performance as a **single number**?
   - Is specifying the mean the correct way?
   - How to report the variability of measured quantities?

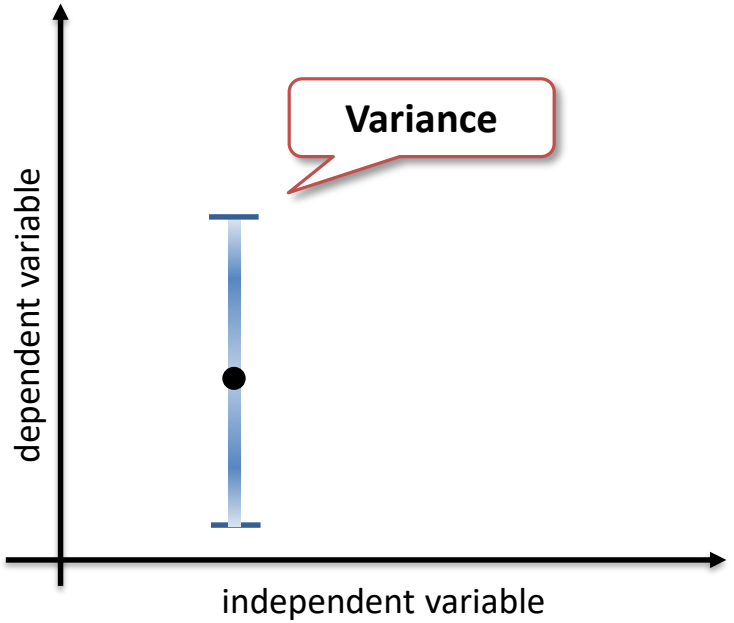




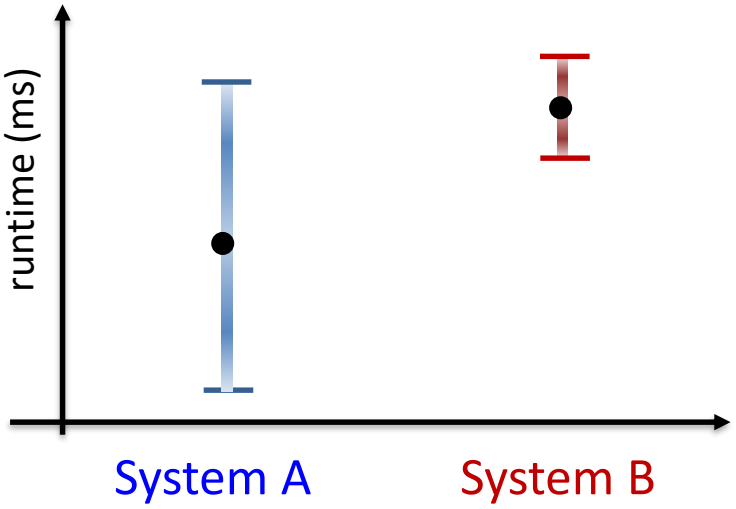
Data summarization

1. How to report the performance as a **single number**?
   - Is specifying the mean the correct way?
   - How to report the variability of measured quantities?





Interpreting results



Question: How do these systems compare?

System A has a less predictable runtime (a lot of runtime variations)
However, in average, it has a smaller runtime than System B.

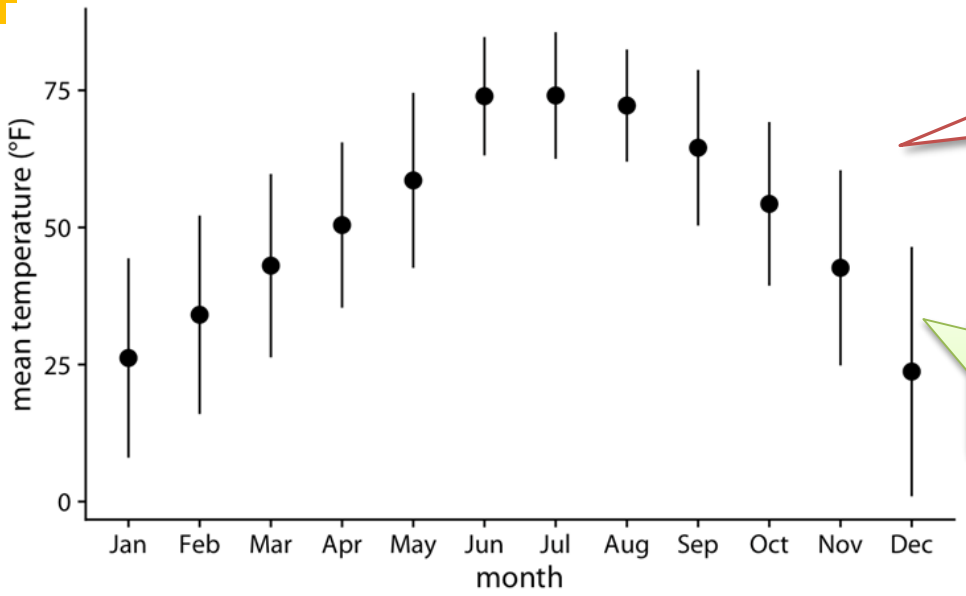
System B has a more predictable runtime, but with a larger average.

Question: Which one do you pick?

Answer: It absolutely depends on the other non-functional requirements of the system, e.g., control systems' performance (stability) degrades if there is jitter, so you prefer system B.

Data summarization

- You can show both mean and variance



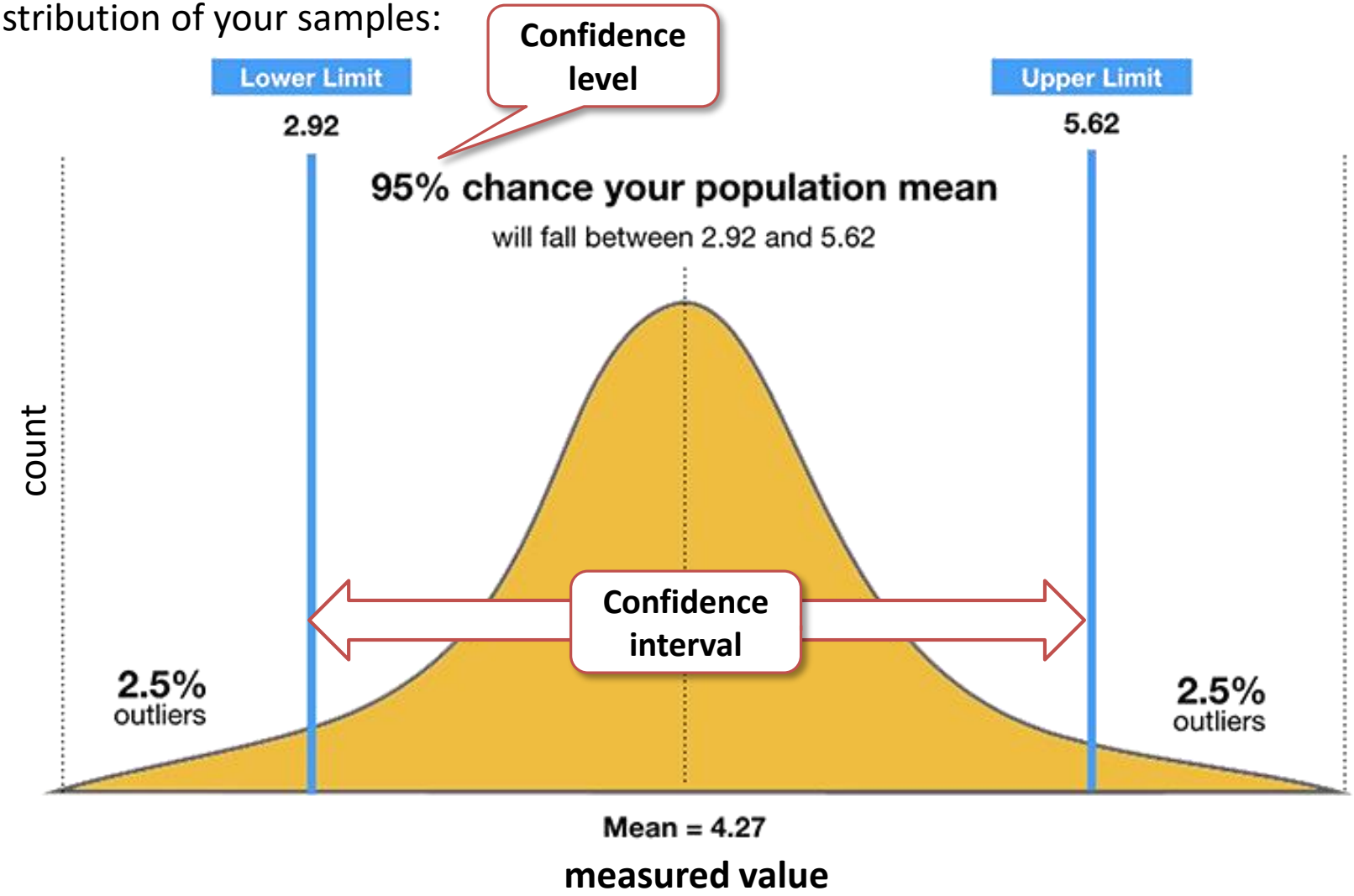
Question: Is it good enough?

Variance does not tell you much about the distribution of data, min, or max

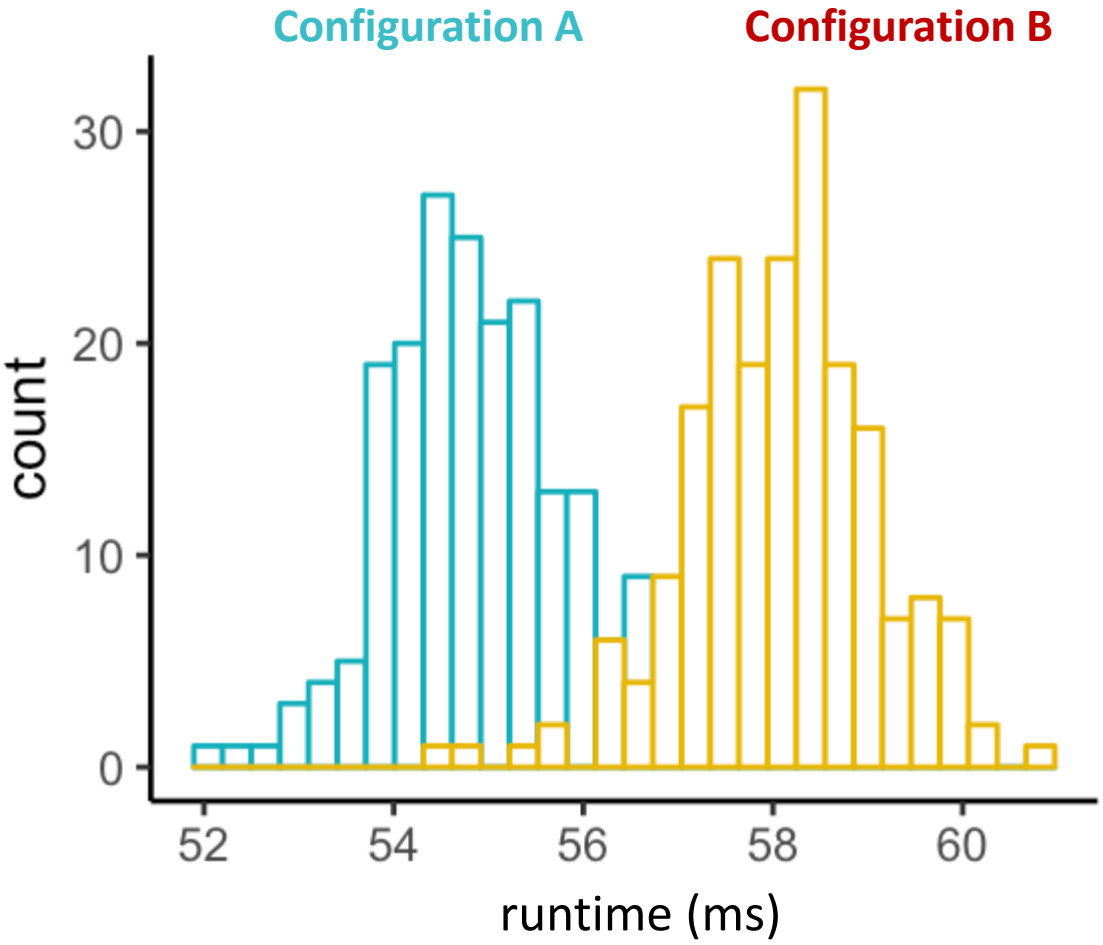
Question: What are the alternatives to variance and when are they appropriate?

Confidence interval

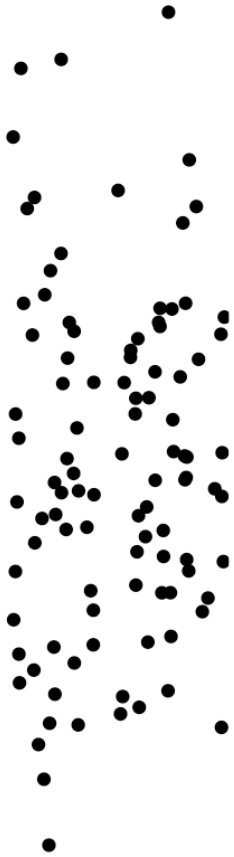
distribution of your samples:

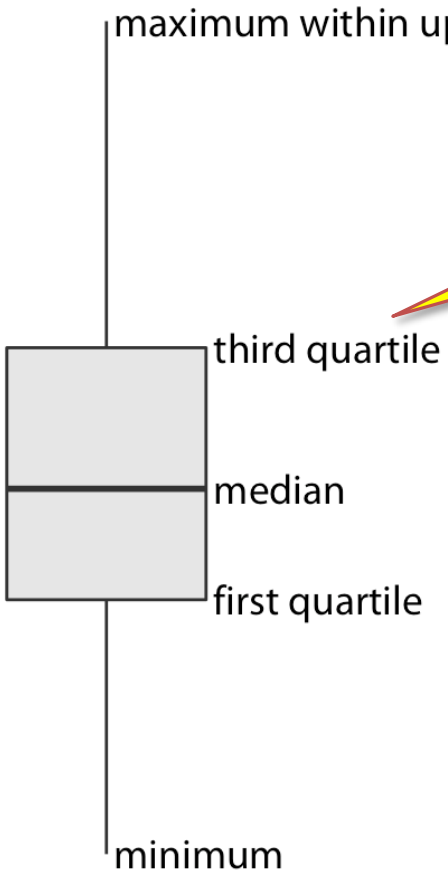


Histogram


Configuration A
Configuration B
30
20
10
0
count
52
54
56
58
60
runtime (ms)

Box plots



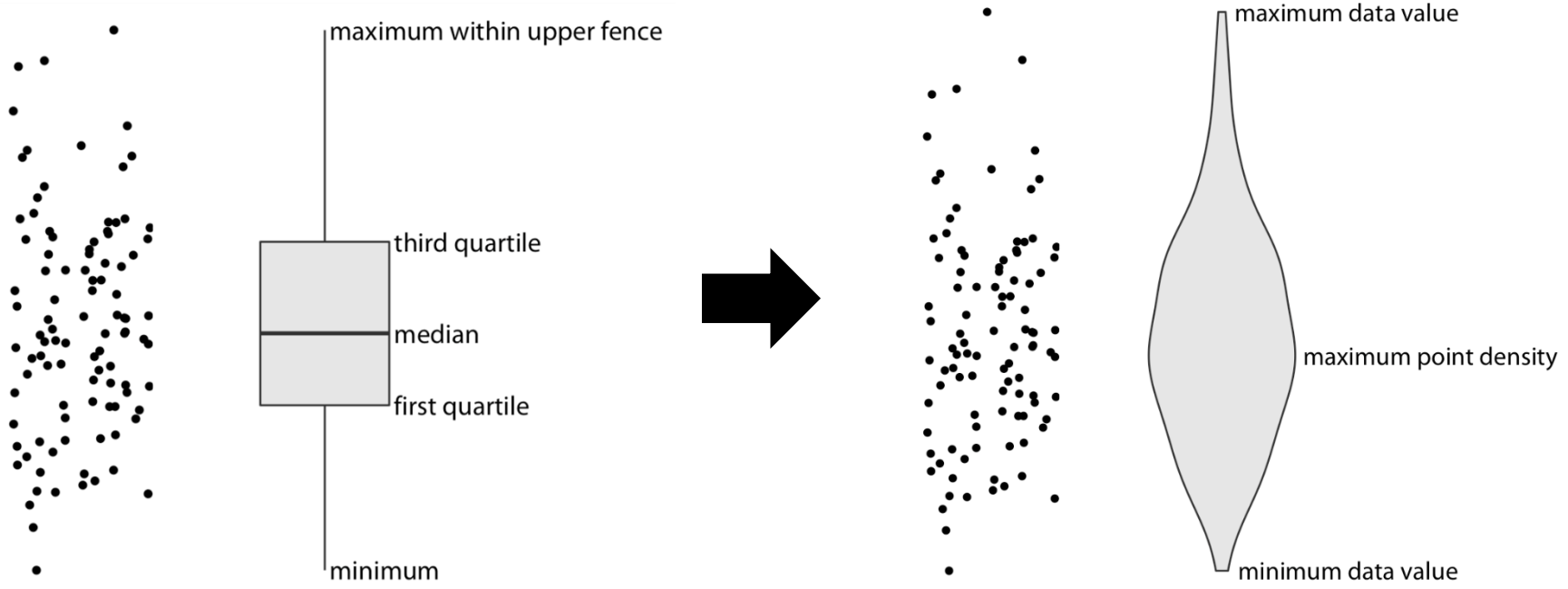


Question: How to obtain this?

Box plot is better than just showing mean and variance

However, it does not show the distribution of your data

Further data representations



source: https://serialmentor.com/dataviz/boxplots-violins.html

Common Mistakes

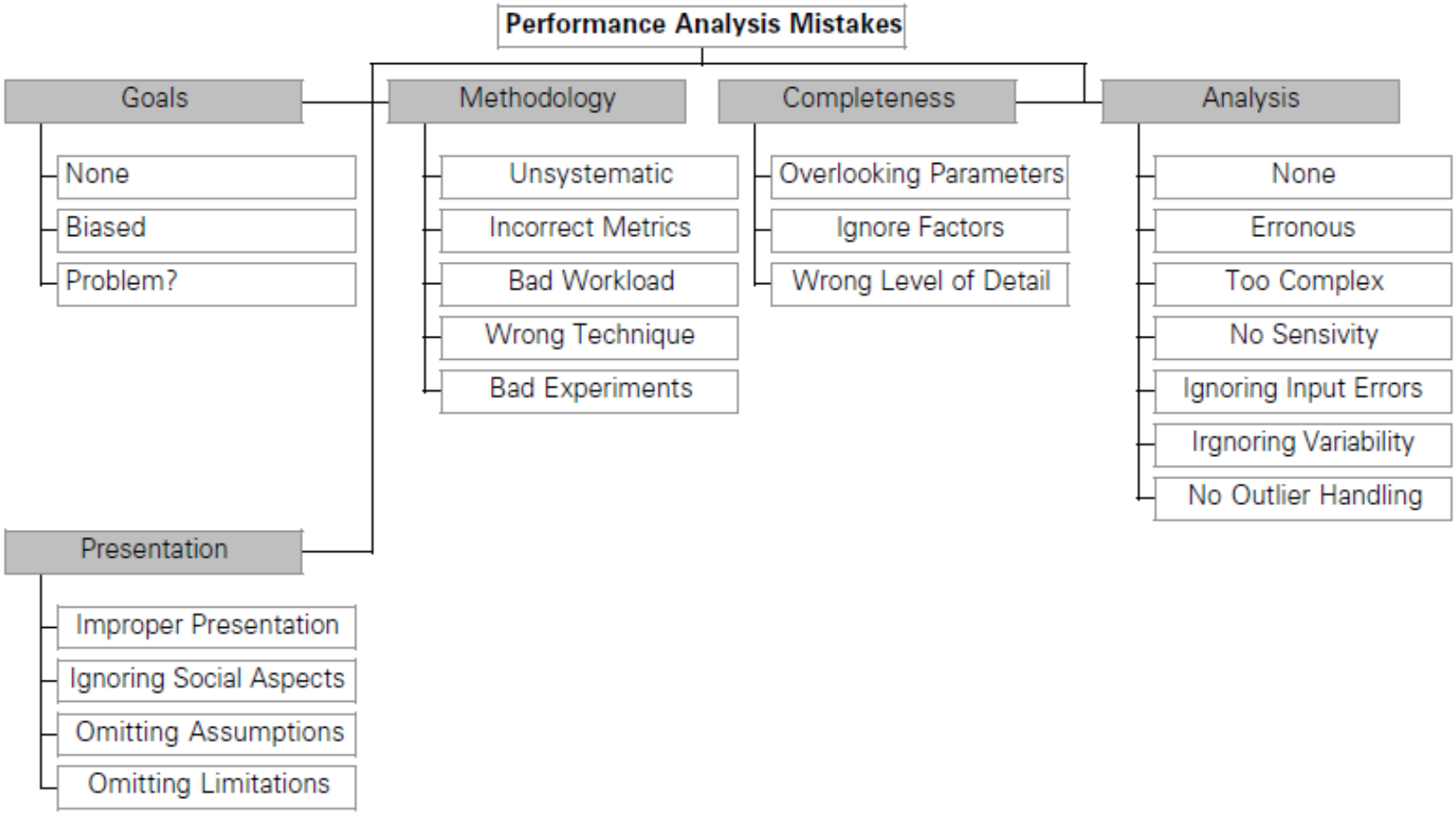
Common mistakes

- **Biased goals**
  - Example:
    - To show that OUR system is better than THEIRS
  - **Mistake:** Finding the metrics and workload such that OUR system turns out better rather than finding the right metrics and workloads.
- **Unsystematic Approach**
  - Selecting system parameters , factors, metrics, and workload arbitrarily
- **Incorrect Performance Metrics**
  - Example: Comparing Two CPUs based on the throughput (MIPS)
    - CISC
    - RISC

Common mistakes

- **Unrepresentative workload**
  - Example: if packet in the network are generally mixture of long and short, workload should consist of short and long packet sizes, not just long ones.
- **Not performing a sensitivity analysis on the factors**
  - Sensitivity analysis answers the question, "if the factors deviate from the expectations, what will the effect be on the system being analyzed".
  - Putting too much emphasis on the results of the analysis, **presenting it as fact rather than evidence.**
- **Ignoring variability of results when reporting the outcome**
  - Only reporting mean value
- **Omitting Assumptions and limitations**
  - Assumption and limitations of the analysis are often omitted in the final report.
  - This may lead the user to apply the analysis to another context where assumption will not be valid anymore.

Common mistakes in performance evaluation

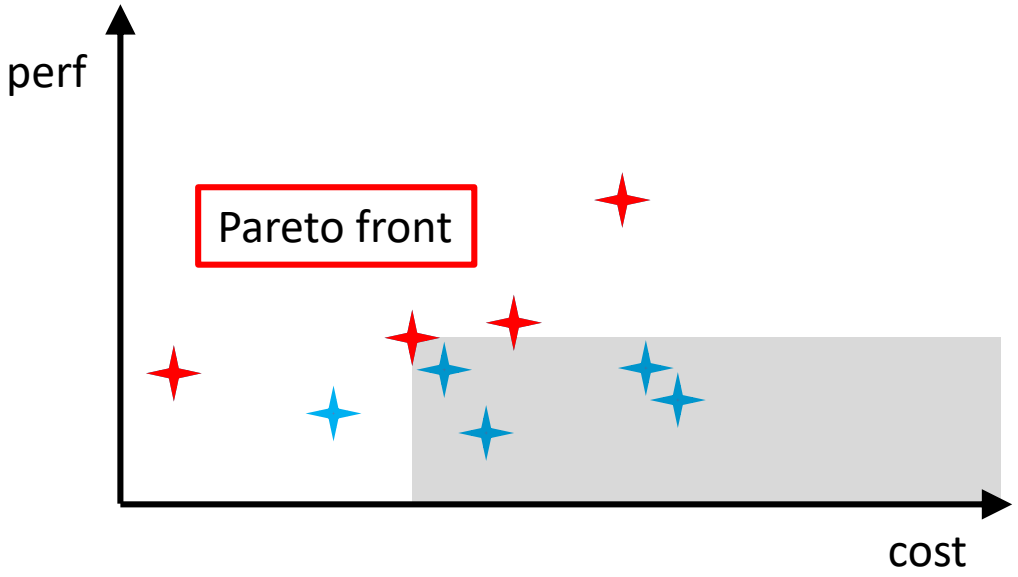


A systematic approach to performance evaluation

1. State goals and define the system
2. List services/functionalities that you want to evaluate
3. Select metrics
4. List workload parameters
5. Select factors and their values
6. Select/build the workload
7. Design the experiment
8. Gather measurement data
9. Analyze the data
10. Present the results

The Emb. Syst. perspective

- ES is about making the right trade off
  - choose the best parameter setting(s), system config, …
- Performance vs. cost

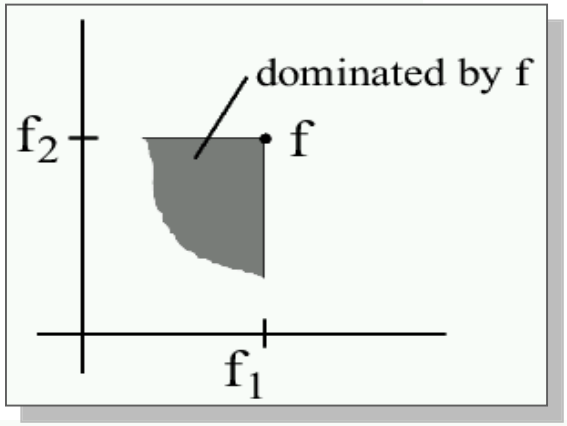


Pareto Optimality

multi-objective optimization

Definition 1 (Dominance relation)
Let $f, g \in \mathbb{R}^m$. Then f is said to dominate g, denoted as $f \succ g$, iff

1. $\forall i \in \{1, \ldots, m\} : f_i \geq g_i$
2. $\exists j \in \{1, \ldots, m\} : f_j > g_j$



Definition 2 (Pareto set)
Let $F \subseteq \mathbb{R}^m$ be a set of vectors. Then the Pareto set $F^* \subseteq F$ is defined as follows: F^* contains all vectors $g \in F$ which are not dominated by any vector $f \in F$, i.e.

$$F^* := \{g \in F | \nexists f \in F : f \succ g\} \quad (1)$$

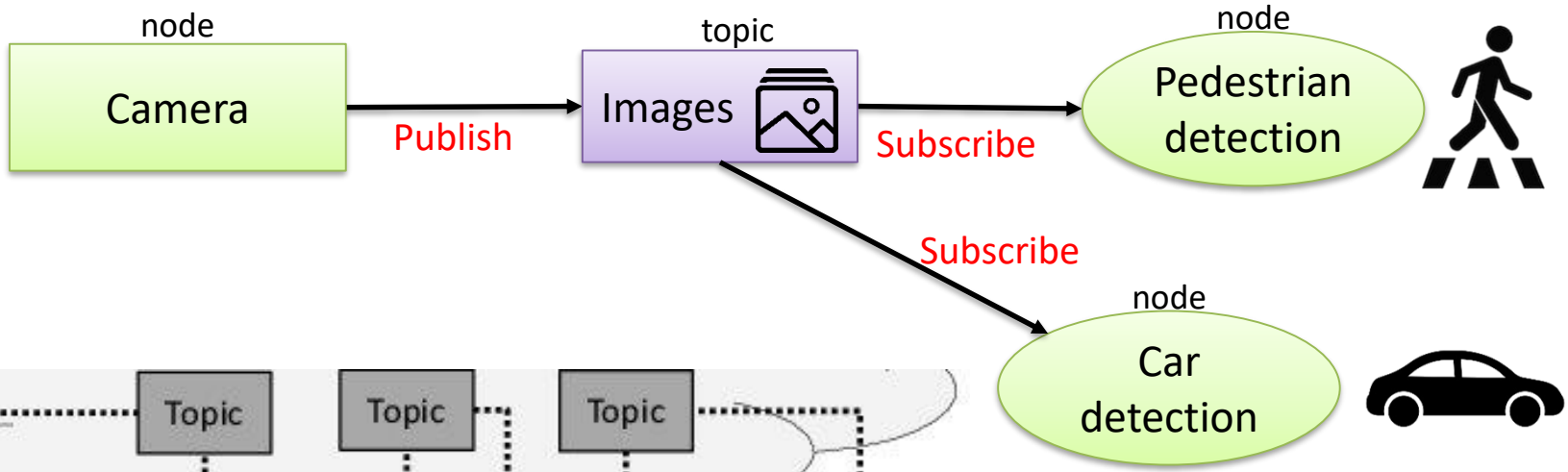
Step 1: open www.kahoot.it in your browser (phone or laptop)
Step 2: enter the pin code and then a nickname

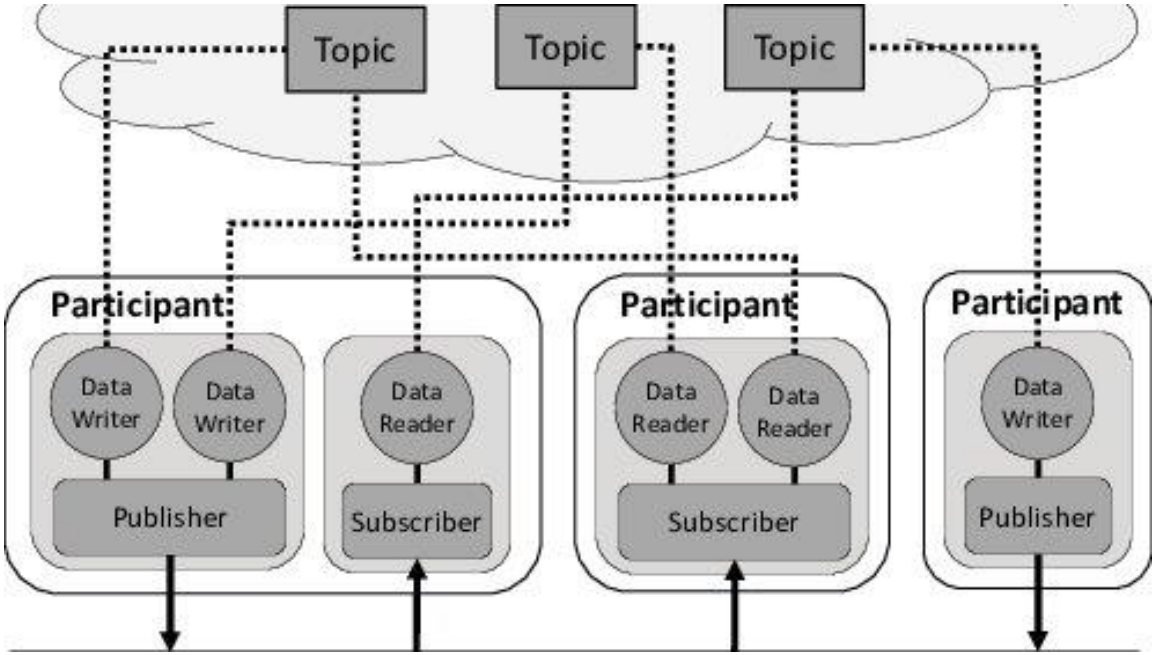
Assignment 1

Assignment 1

- Is focused on evaluating ROS 2.0's communication overheads
- ROS 2.0 is the latest generation of ROS (Robotics operating system)
- The assignment will be released Monday
- Lab starts from tomorrow ☺

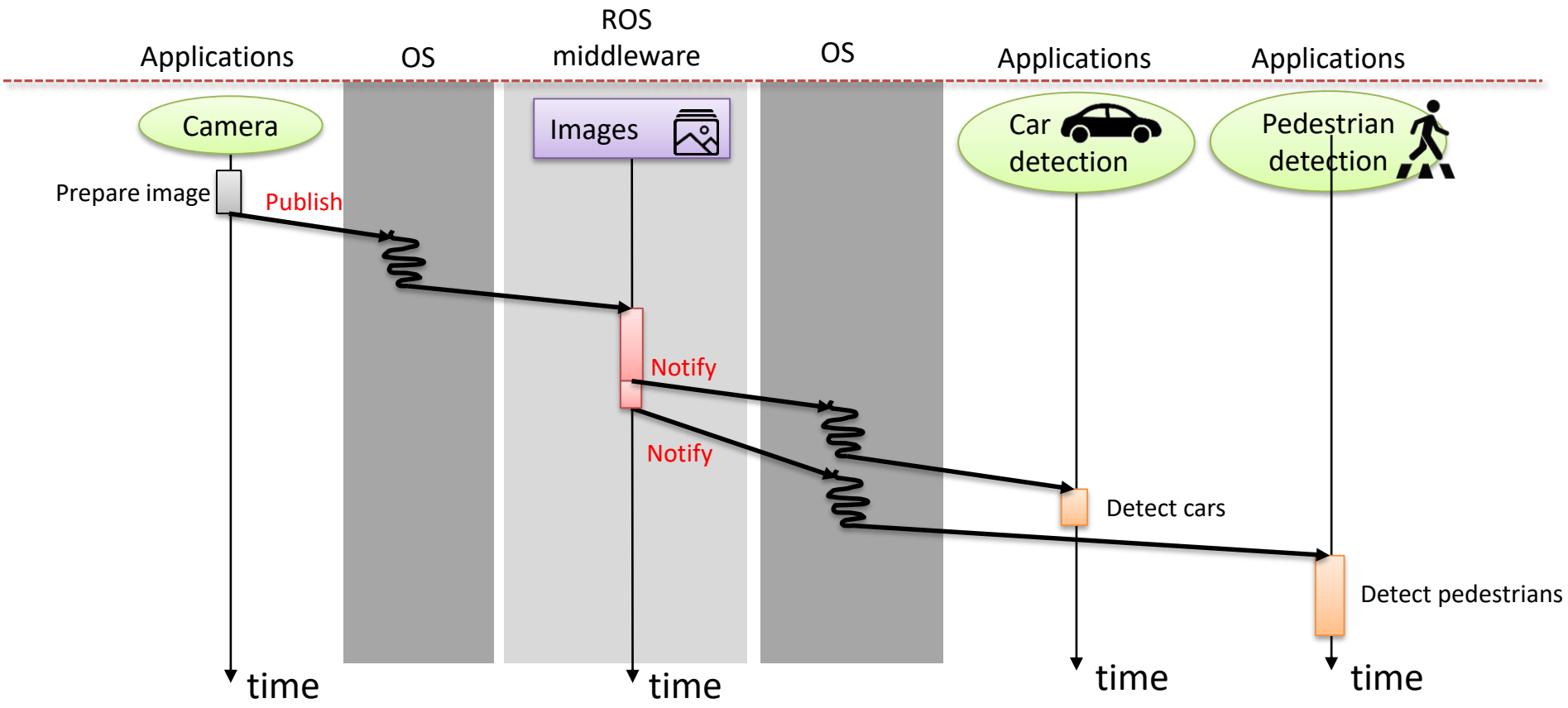
ROS 2.0 – example



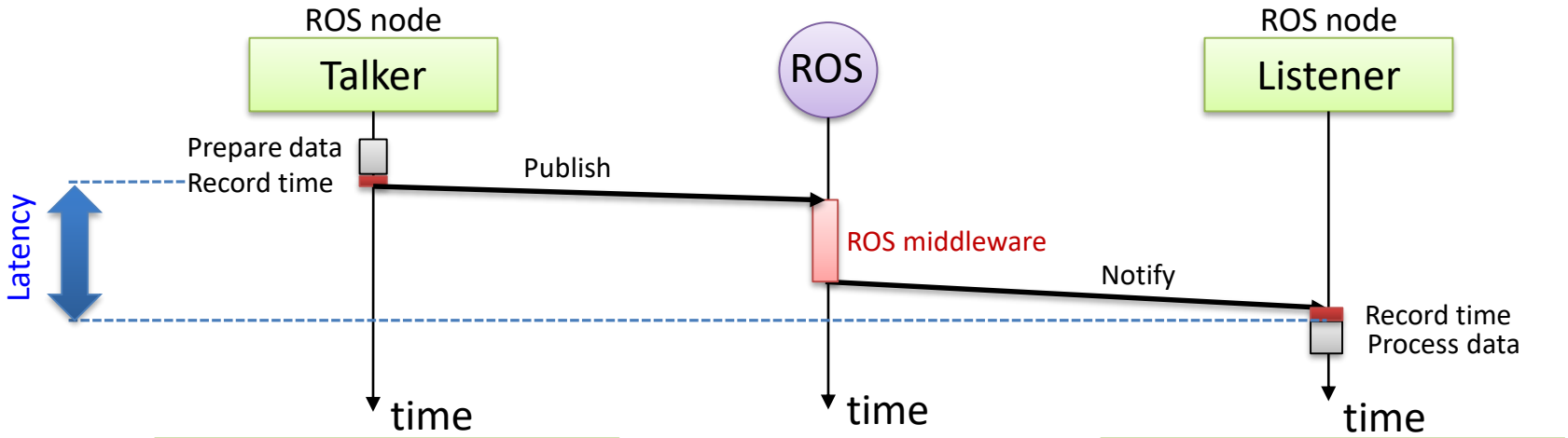


As soon as a data is sent to a topic, ROS's executor is activated and calls the call back of the subscribers.

ROS 2.0 – under the hood



Case study for Assignment 1



Each time stamp is stored at *publish_time_XKbyte.txt*

(where X shows the data size).

Each time stamp is stored at *subscribe_time_XKbyte.txt*

(where X shows the data size).

- **Factors**
  - ROS configurations
    - 3 DDS choices: FastRTPS, OpenSplice, Connext
  - OS configurations
    - process priority
    - Smart CPU frequency scaling: active, inactive
  - Size of message
    - 4KB, 16KB, …, 4MB (in total 8 values)

Questions?

Sources used to make the slides

- Jean-Marc Vincent and Arnaud Legrand
  - Performance measurements of computer systems: tools and analysis, M2R PDES, 2015
  - http://polaris.imag.fr/arnaud.legrand/teaching/2015/M2R_EP_measurements.pdf
- Holger Brunst and Matthias S. Mueller
  - Performance Analysis of Computer Systems
  - Center for Information Services and High Performance Computing (ZIH), TU Dresden
  - https://tu-dresden.de/zih/ressourcen/dateien/lehre/ws1112/lars/vorlesungen/lars_lecture_02_requirements-metrics-techniques.pdf?lang=en
- William Gropp
  - Course: Designing and Building Applications for Extreme Scale Systems, Illinois, 2016
  - http://wgropp.cs.illinois.edu/courses/cs598-s16/index.htm
- Raj Jain
  - The Art of Computer Systems Performance Analysis
  - https://www.cse.wustl.edu/~jain/books/perf_sli.htm
- Other slides and sources:
  - https://slideplayer.com/slide/7709208/
  - https://www.slideshare.net/ShubraBansal/uncertainity-in-measurement
  - https://serialmentor.com/dataviz/boxplots-violins.html

Thank you