Performance Metric Properties

Vojtěch Horký Peter Libič Petr Tůma

2010 – 2021

Contents

1 Overview

Performance Metric

A performance metric is a quantitative measure of some property of interest.

Typically, they are one of:
- Count of an event of interest,
- Duration (interval between events),
- Size (value) of a parameter of interest.

We are not that much interested in formal properties. But we still have many practical requirements.

Properties of Interest

Speed
- System completes the task successfully and provides correct results.
- We are interested in how fast it performs the task.

Efficiency
- System completes the task successfully and provides correct results.
- We are interested in how many resources were used.

Reliability
- System completes the task but the result is incorrect.
- We can measure how often the errors happen.

Availability
- The system did not perform the task because it was down.
- We can measure how much the system is (not) available.

2 Practical Requirements

Good Performance Metric

What is a good performance metric ?

Goals

Comparing two computer systems ? Evaluating an optimization ? Estimating execution cost ?

Audience

Developers ? Researchers ? Customers ? Private or public ?

Dangers

Poorly chosen metrics can be misleading !

- Hard to interpret.
- Leading to incorrect conclusions.
- Measuring features that are not interesting.

Practical Requirements for Good Metric

A good metric should be:

- Linear.
- Reliable.
- Repeatable.
- Easy to measure.
- Consistent.
- Independent.

These goals cannot always be met and can be contradictory. But it is good to get close.
[1]

Requirement: Linearity

Why ?

Linear metrics are easier for humans to interpret.

- If a metric doubles its value, the system should be twice as fast, or finish the task in half the time.
- Linearity is not met by many metrics:
  - Well known example is dB (acoustic pressure).
  - Also camera resolution vs image dimensions.
  - Or cache size vs miss rate or speed up.
  - They are not wrong, but may be much harder to interpret.

> Typical rule of thumb is 10 dB is twice as loud.

Requirement: Reliability

Why ?

We expect better values indicate better systems.

- One system outperforms another when the metric values indicate so.
- Many reasonable examples:
  - Network bandwidth, copying files over faster network should be faster.

[1] Based on Lilja: Measuring Computer Performance ... doi:10.1017/CBO9780511612398

  - Memory speed, running on faster memory should be faster.
  - But what about processor clock speed ?
  - Or cost ?
- Hard to guarantee for very general metrics (performance is application specific).
- Quite obvious, but often not met !

Is clock speed a reliable metric ? And does it depend on context ?

Requirement: Repeatability

Why ?

We expect the metric to be an inherent system property, hence repeatable.

- Each run of an experiment should give the same value of a metric.
- Not completely realistic:
  - Computers are not always deterministic (randomized algorithms, asynchronous interrupts ...).
  - Full control over experiment not always possible (distributed systems, database servers, cloud ...).
  - Statistical methods can help attribute variability.
  - Metric can be deterministic, thus repeatable (number of instructions in a program repeatable but not reliable).

Requirement: Ease of Measurement

Why ?

Obviously, if we cannot get metric values it is a problem ...

- A metric should be easy to measure or infer from other (easily measurable) metrics.
- More difficult to measure means more likely measured incorrectly.
  - One way network latency.
  - Single thread power consumption.
  - Timing of synchronization construct in code (measurement difficulty not strictly property of metric).

One Way Network Latency

Think how difficult it is to measure network roundtrip time vs (one way) network latency. Would synchronized clock help ? And how to validate clock synchronization ?

Requirement: Consistency

Why ?

Same meaning everywhere facilitates system comparison.

- A metric should have the same units everywhere.
- The units should have the same meaning everywhere.
- Metrics like MIPS or MFLOPS do not follow this obvious requirement.

Requirement: Independence

Why ?

Trust in metric requires system (and hence vendor) independence.

- Systems should not be optimized for particular metric.
- But vendors are known to optimise for specific benchmark (metric) !
  - For example nVidia and 3DMark, Sun and SPECjbb2000.
  - This makes evaluation results less representative.
- But...
  - Developers need benchmarks to test and optimize their code.

- For compilers, SPEC CPU seems to be a good set, tries to be representative.

Even an initially independent metric can become an optimization target !

What Exactly Is Cheating ?

In 2003, nVidia graphics drivers optimized some processing in a way that benefited the 3DMark benchmark. The benchmark authors initially considered this cheating, but eventually concluded that it is an application specific optimization instead. https://www.extremetech.com/computing/54154-driver-irregularities-may-inflate-nvidia-benchmarks https://www.extremetech.com/extreme/54318-update-futuremark-now-says-nvidia-didnt-cheat

In 2009, Intel graphics drivers moved some processing from GPU to CPU when the workload saturated the GPU. The workload was recognized by executable name, and apart from games the list included the 3DMark benchmark. https://techreport.com/review/17732/intel-graphics-drivers-employ-questionable-3dmark-vantage-optimizations

In 2013, some Samsung devices selected between faster and slower cores depending on whether certain executables were running, the list of executables included benchmars. https://www.anandtech.com/show/7187/looking-at-cpugpu-benchmark-optimizations-galaxy-s-4

In 2018, several Huawei devices were observed to overclock beyond standard power and thermal limits when benchmark executables were running. https://www.anandtech.com/show/13318/huawei-benchmark-cheating-headache

3 Metric Selection

Selecting Metrics For an Experiment

1. List all metrics possibly measurable in the given scenario.
2. Select a reasonable subset following these criteria:
   - Low variability Helps to reduce number of required repetitions. Computing ratio usually increases variance, better avoid.
   - Non redundancy If one metric can be derived from another, choose only one.
   - Completeness Try to select so many metrics that all possible outcomes are included.
   - Insight Choose metrics that provide insight or validate hypotheses.
3. While executing experiments, watch for anomalies, extend observed metrics, then repeat.

[2]

[2] Based on Jain: The Art of Computer Systems ... ISBN 978-0-471-50336-1