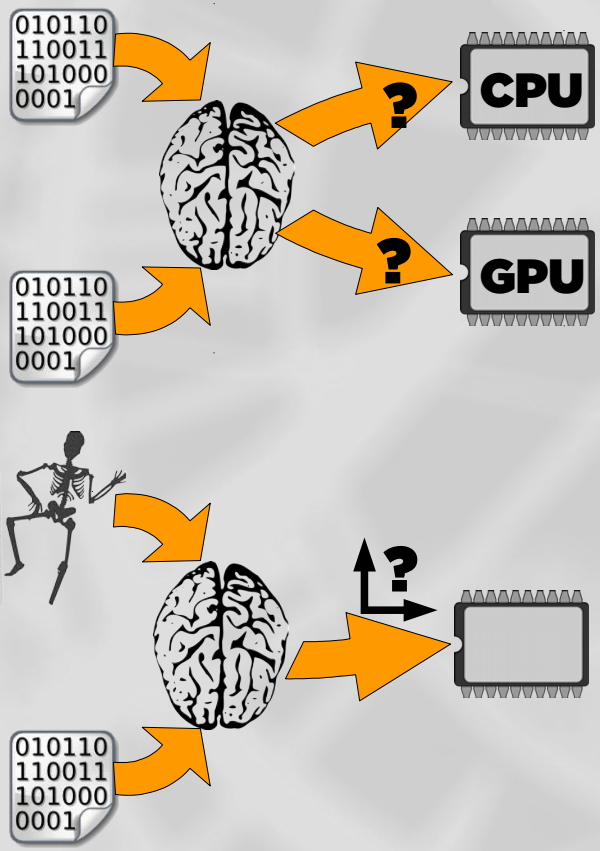
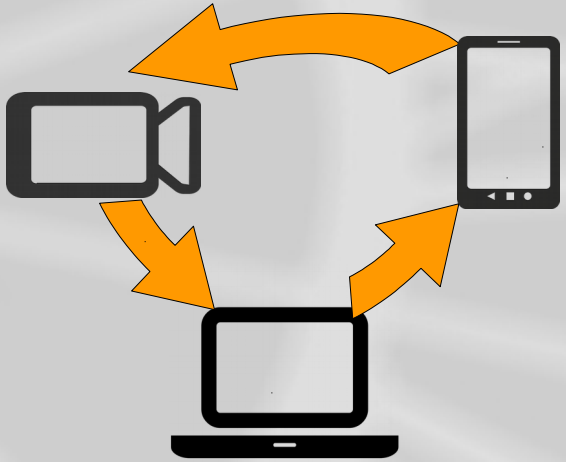
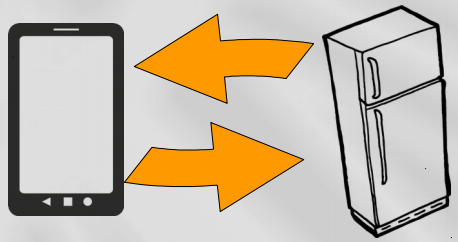
How fast?
How furious?

Real people
Real optimizations

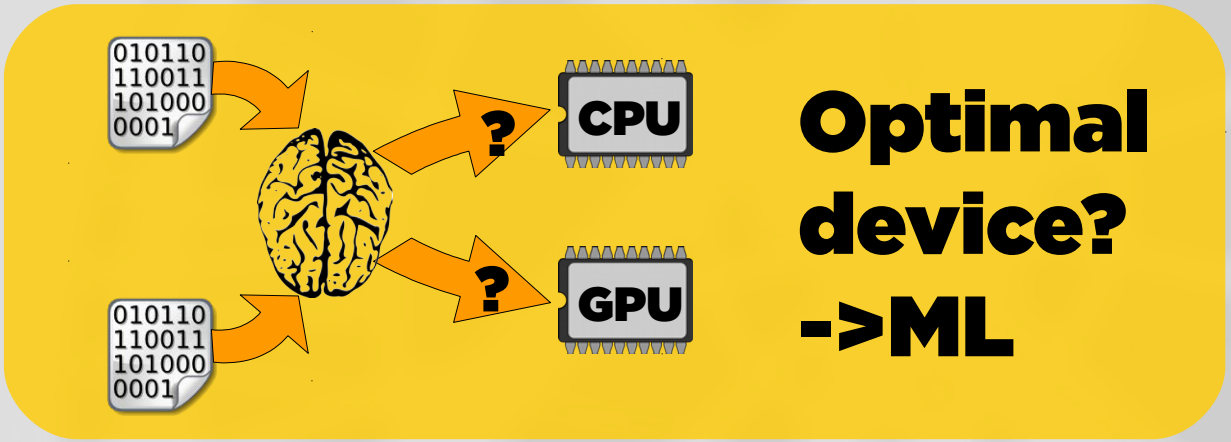
Research

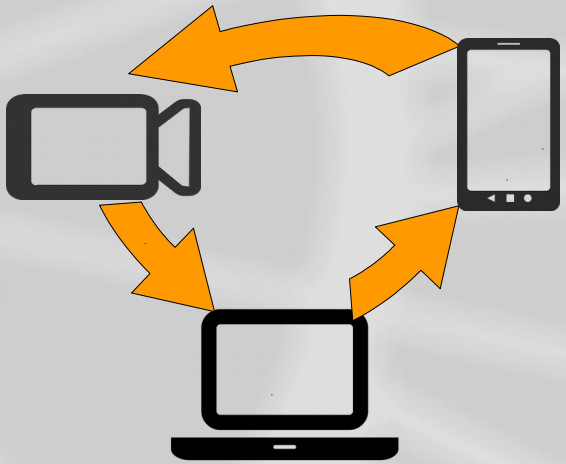

010110
110011
101000
0001
?
CPU
010110
110011
101000
0001
?
GPU
?
010110
110011
101000
0001

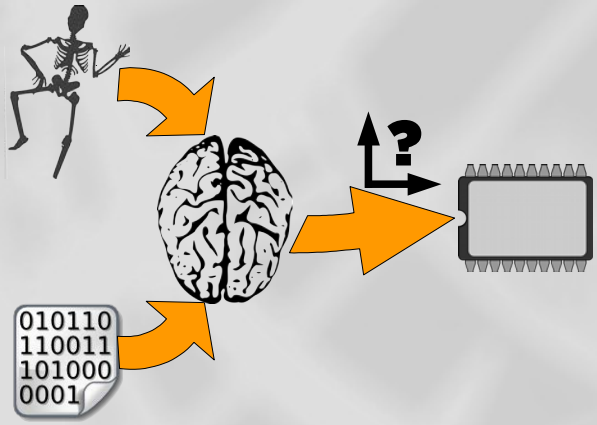


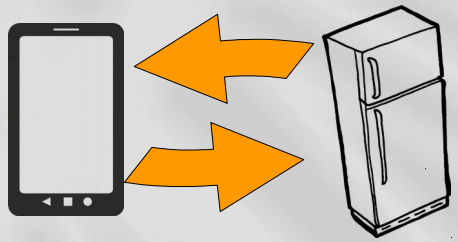


Research

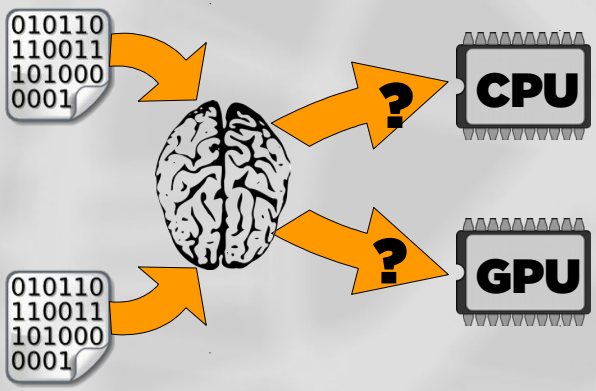

010110
110011
101000
0001
?
CPU
?
GPU
010110
110011
101000
0001
Optimal device? ->ML

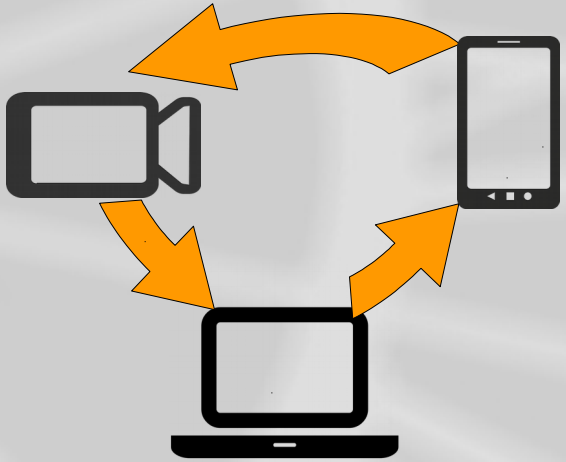


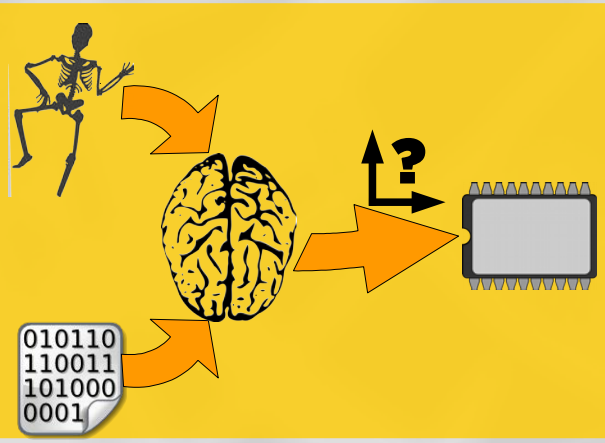

?
010110
110011
101000
0001



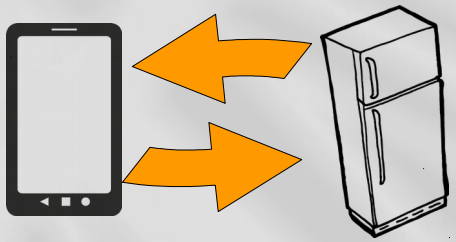
Research



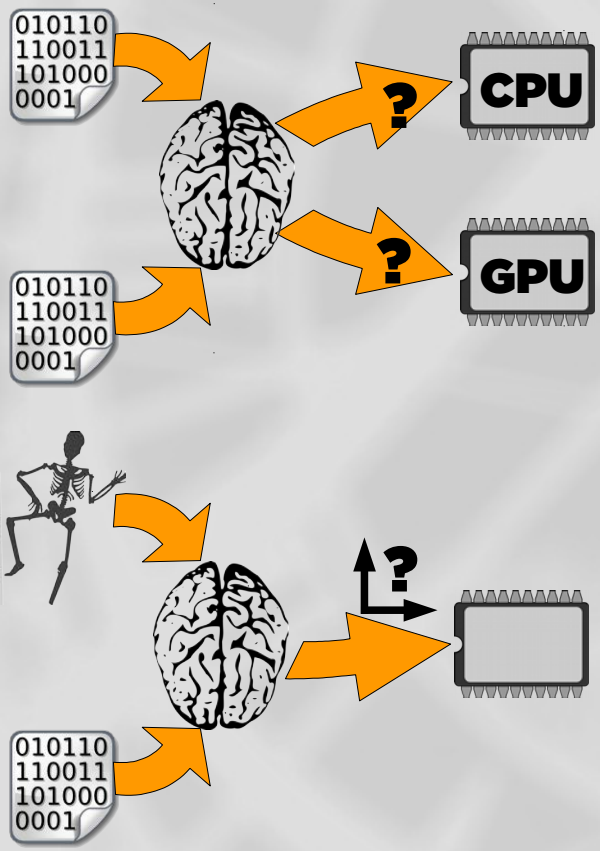




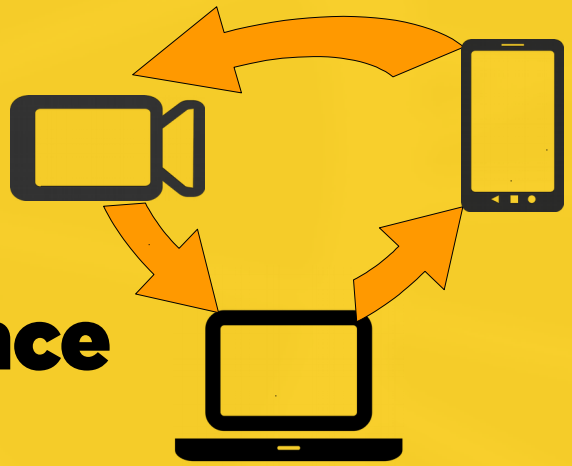
OpenCL workgroup size?

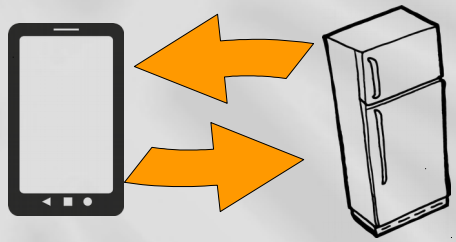


Research

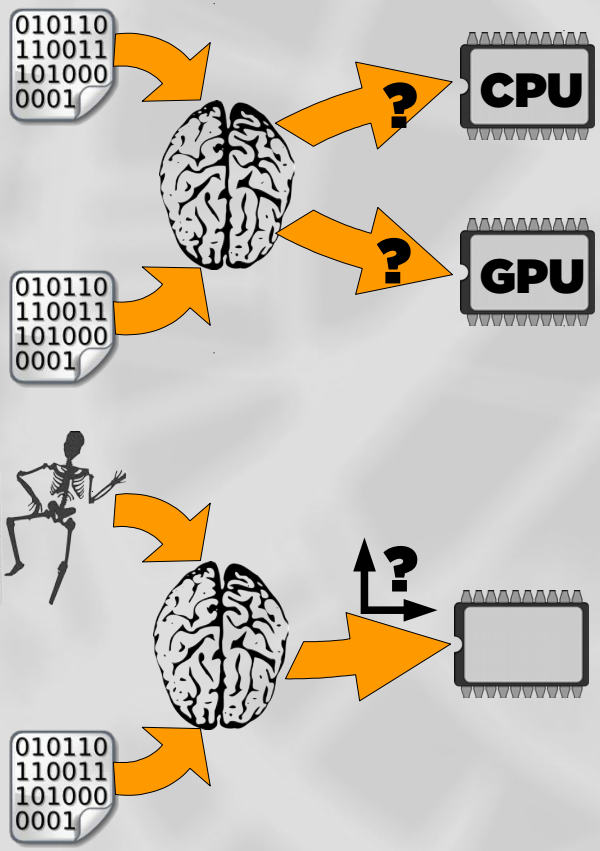


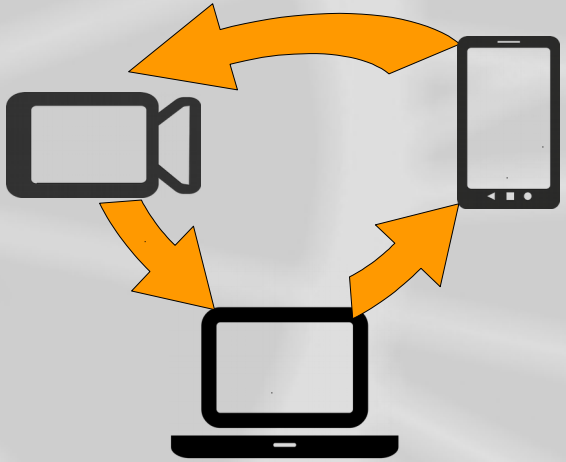
Quality Of Experience



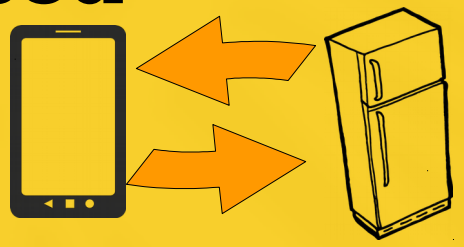


Research

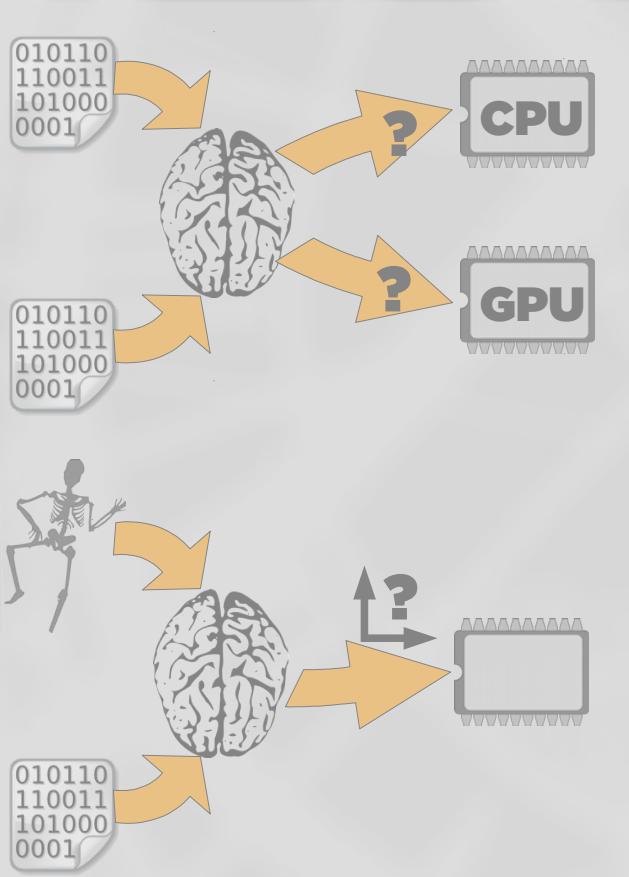




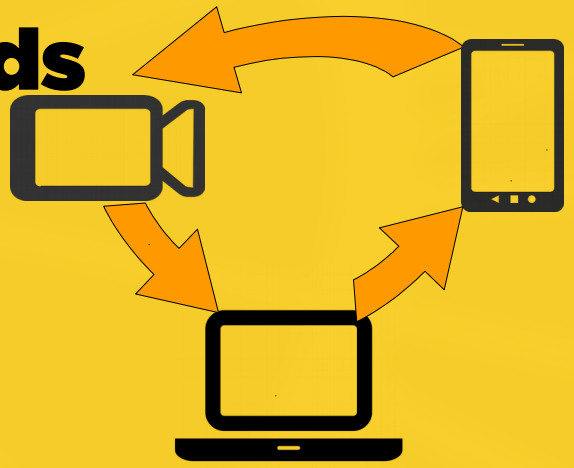
Personalised optimi-sations for smartphones



Research

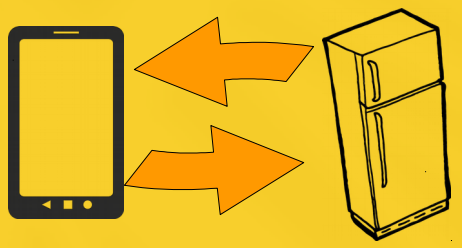


Real workloads



Real metrics

Real people



People

Hugh Leather

Volker Seeker

Paschalis Mpeis

Quality of Experience

Testing
Evaluating
Tuning

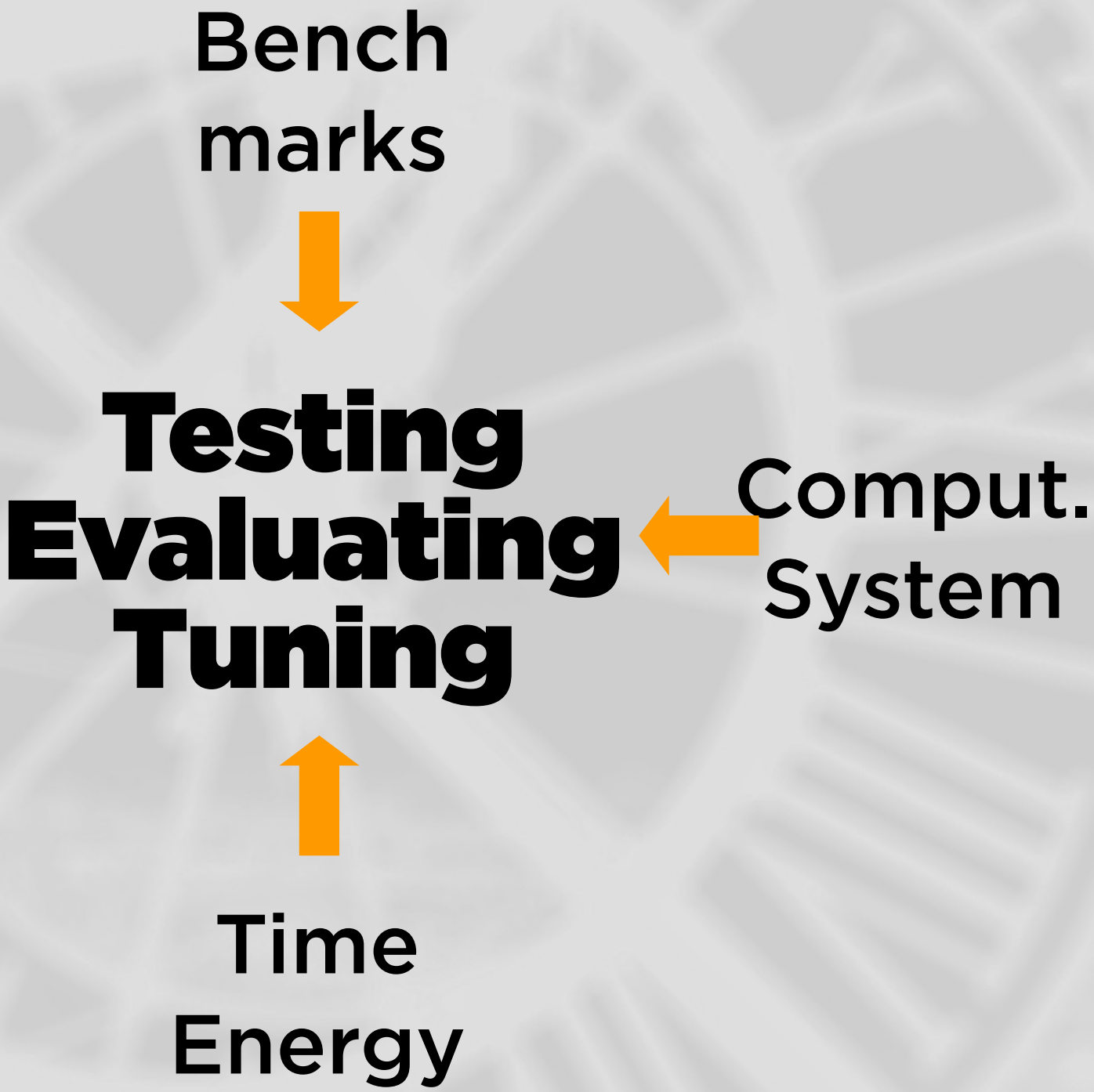
Bench marks

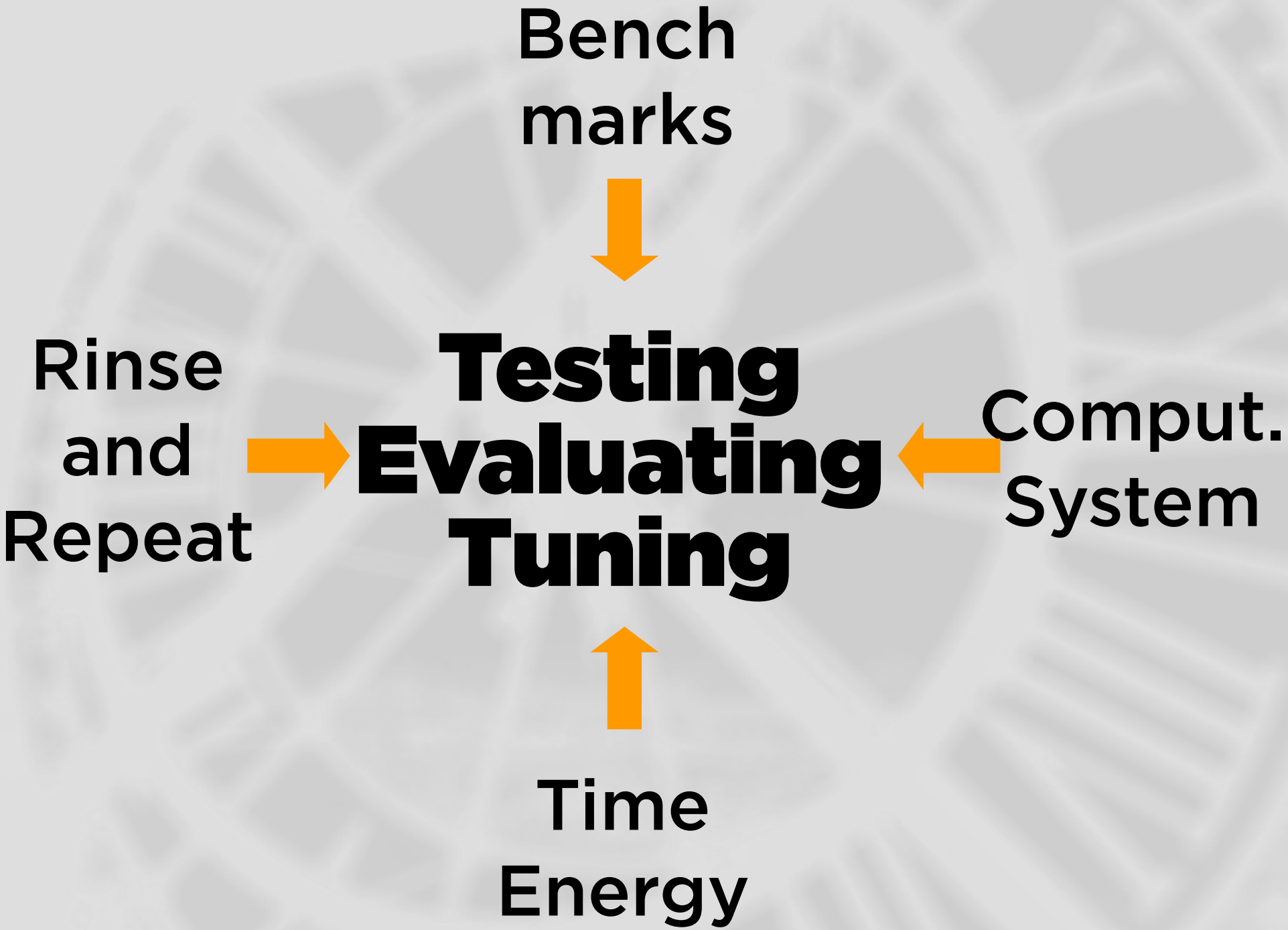
Testing
Evaluating
Tuning

Bench marks

Testing
Evaluating
Tuning

Comput. System

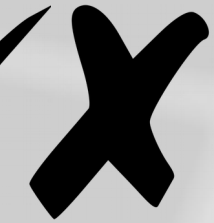

Bench marks
Testing
Evaluating
Tuning
Comput. System
Time
Energy


Bench marks
Rinse and Repeat
Testing
Evaluating
Tuning
Comput. System
Time
Energy

One size fits all?

Servers ✓

One size fits all?

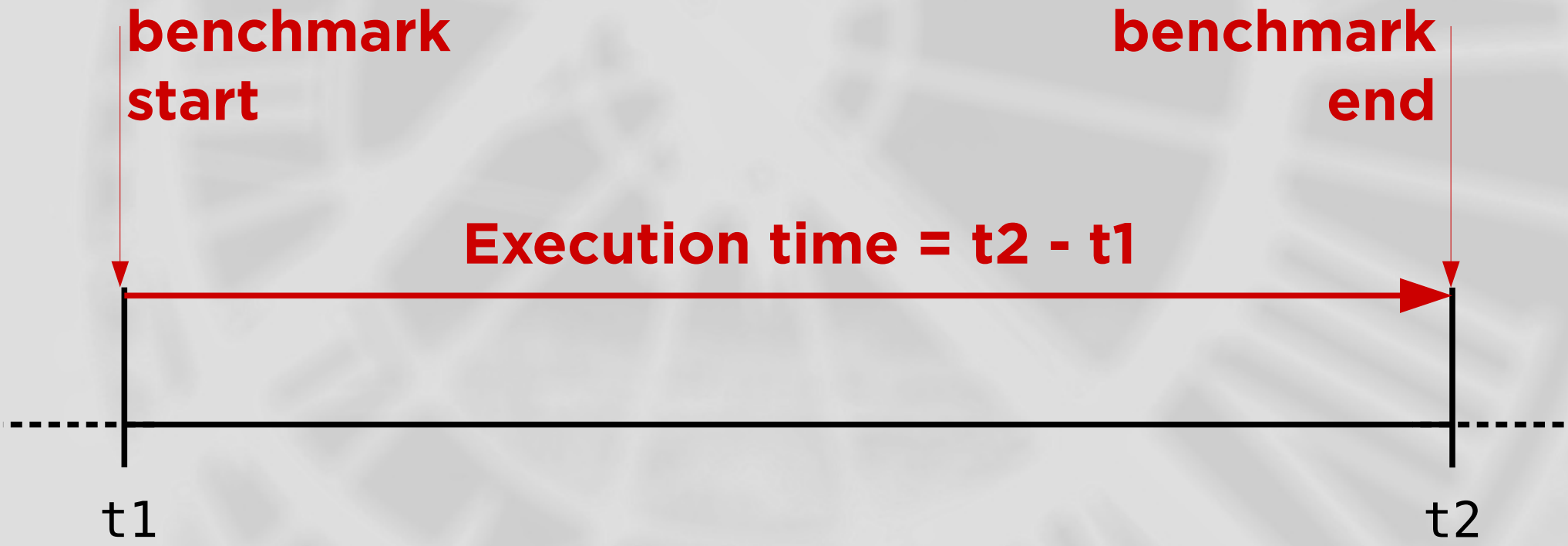
Desktops 

One size fits all?

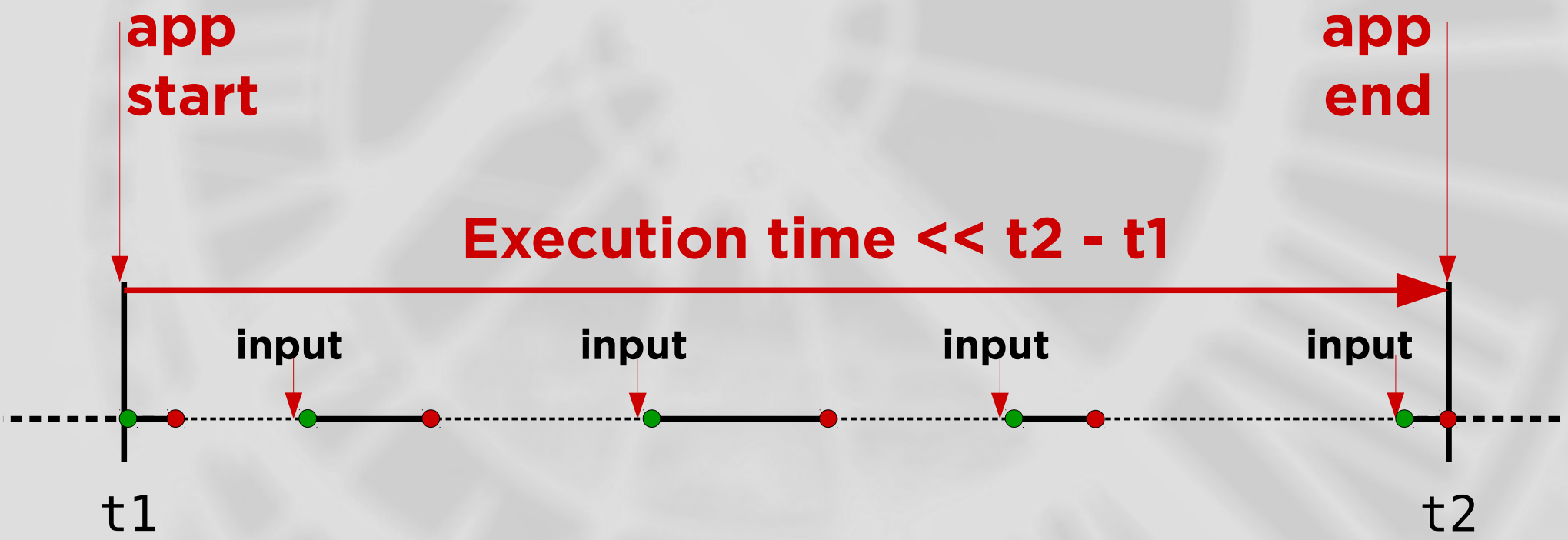
Smartphones 

What does the user really care about?

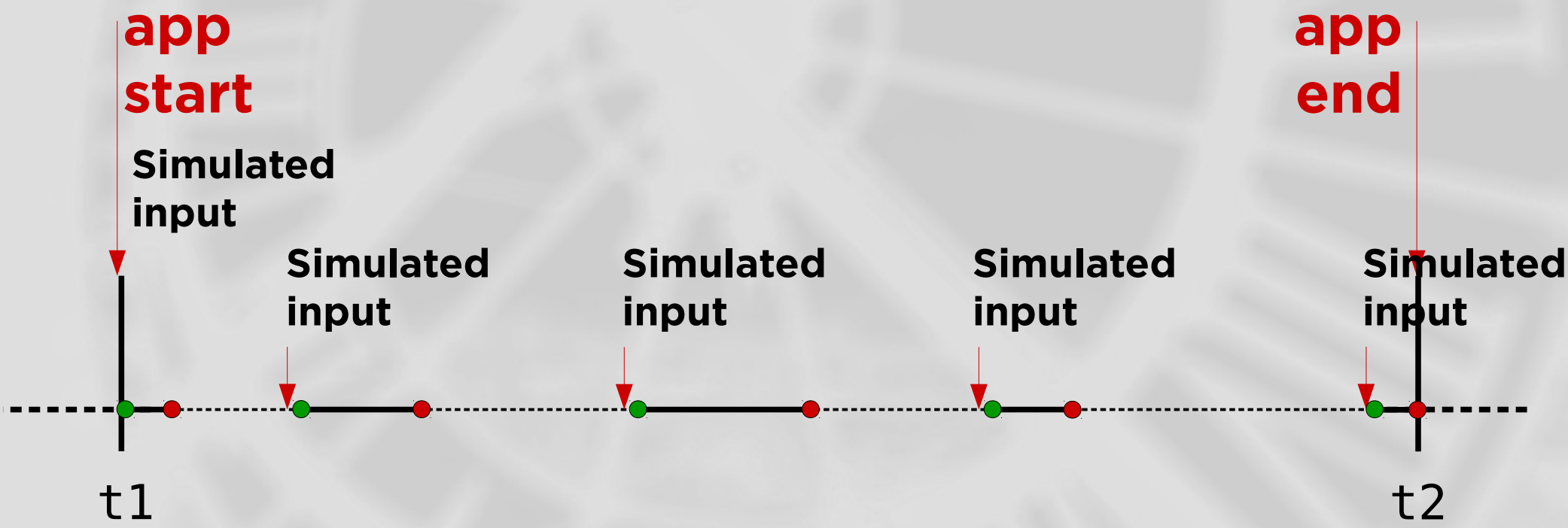
Standard Benchmarks



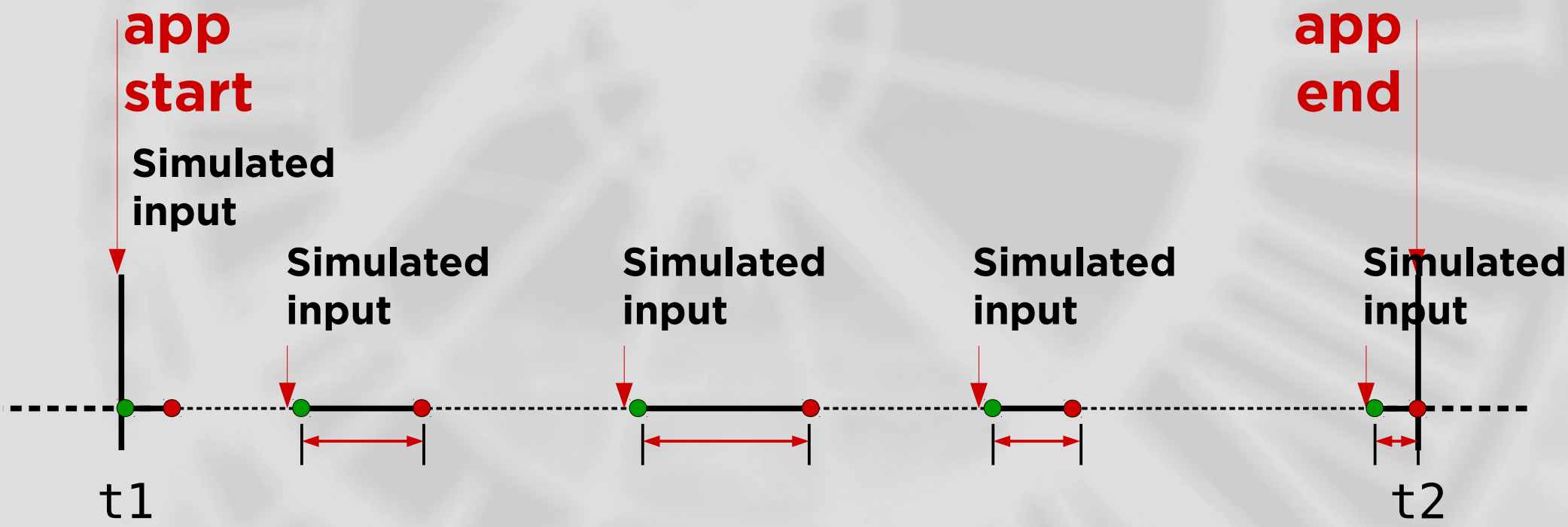
Mobile apps


app start
app end
Execution time << t2 - t1
input
input
input
input
t1
t2

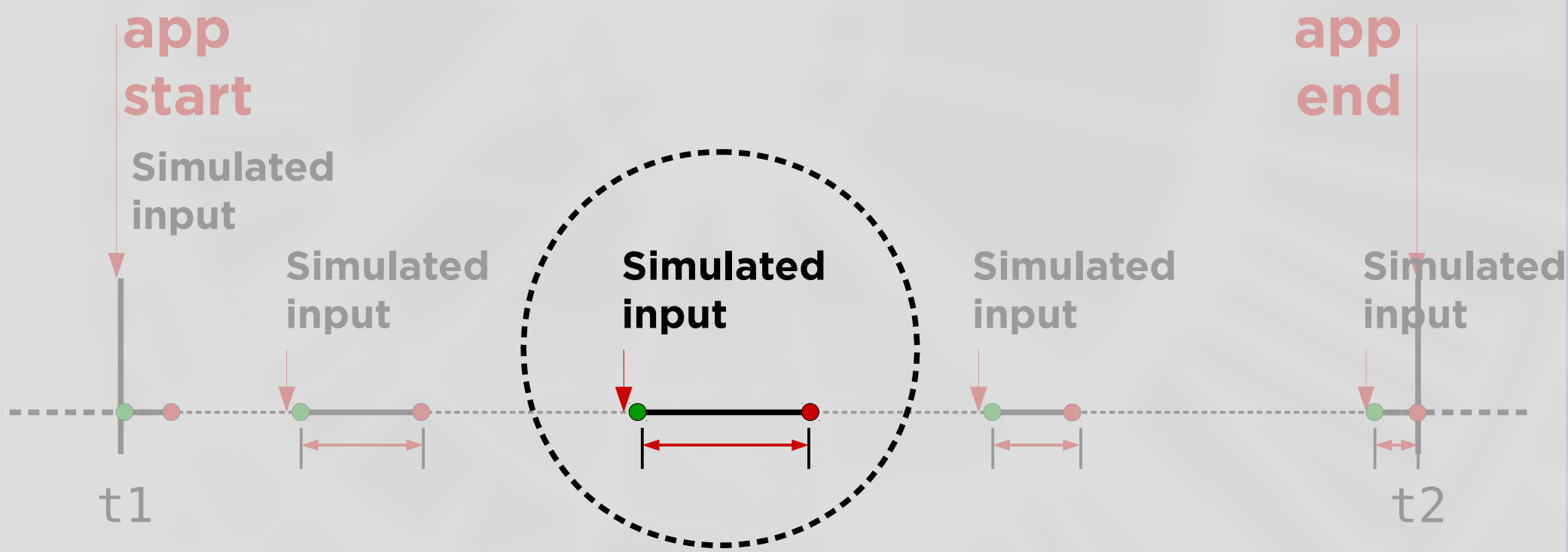
Input Replay



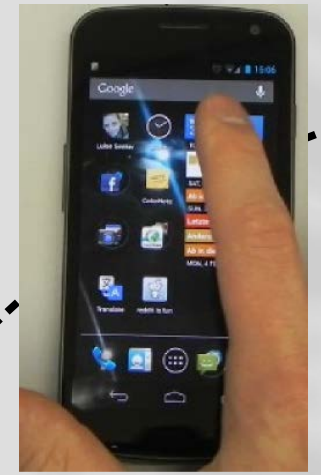
Input Replay



Input Replay

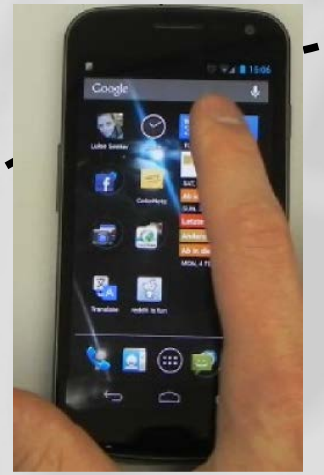


What does the user care about?



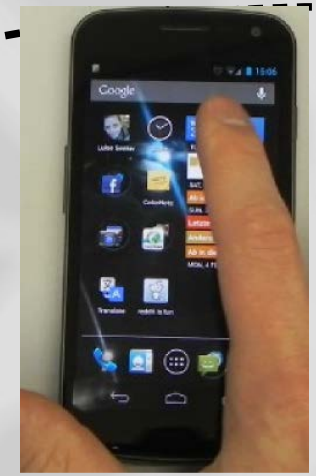
input

What does the user care about?



I/O

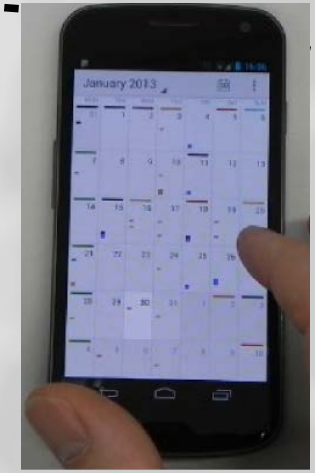
What does the user care about?



I/O

Processing

What does the user care about?

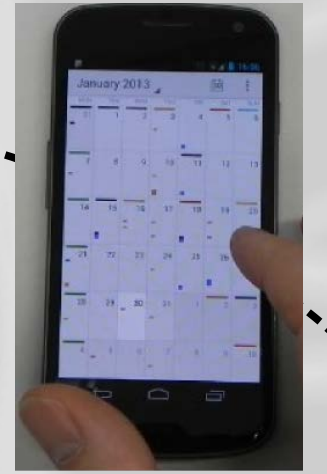


I/O

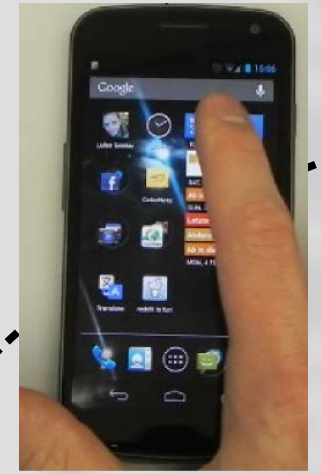
Processing

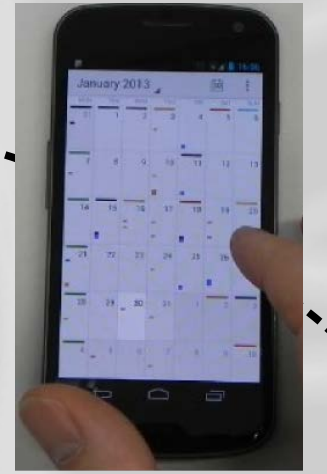
Screen Update

What does the user care about?



What does the user care about?

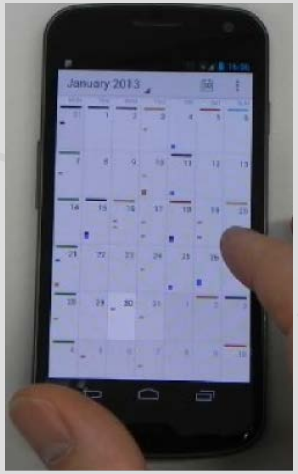




System's perception of latency

User's perception of latency

What does the user care about?



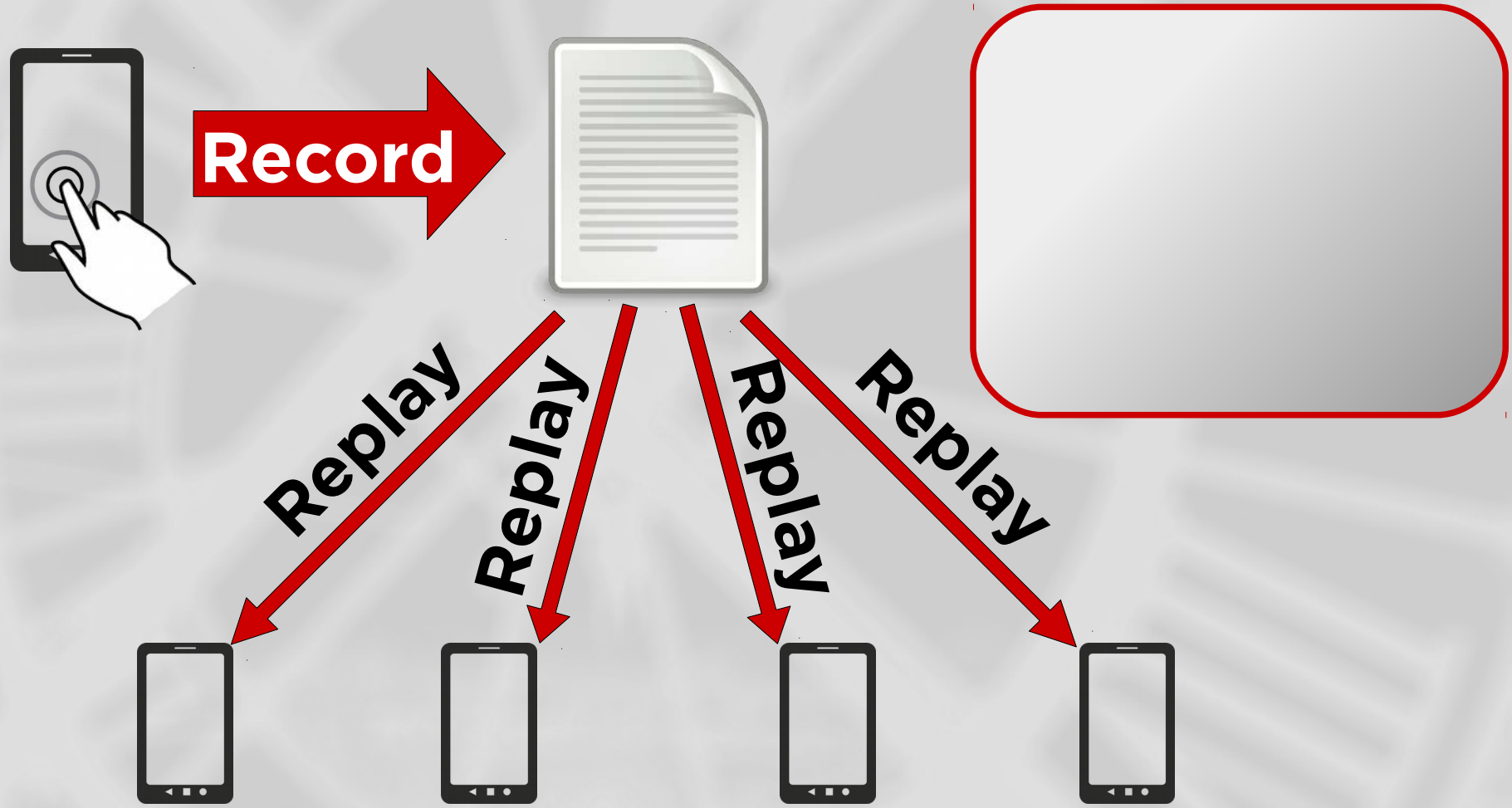
Interaction Lag

Quantifying interaction lags

Record/Replay

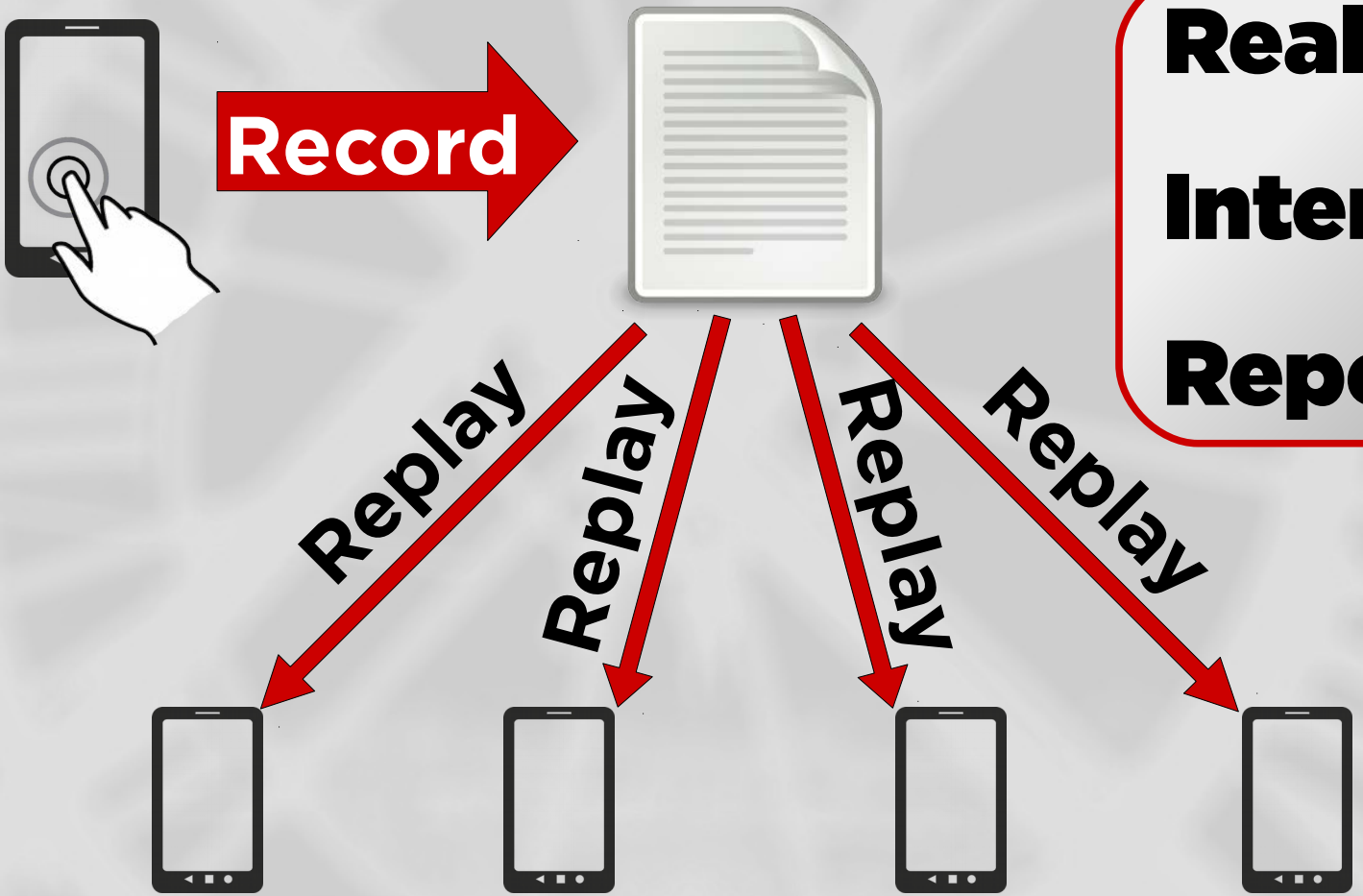
Sample

Record/Replay



Sample

Record/Replay



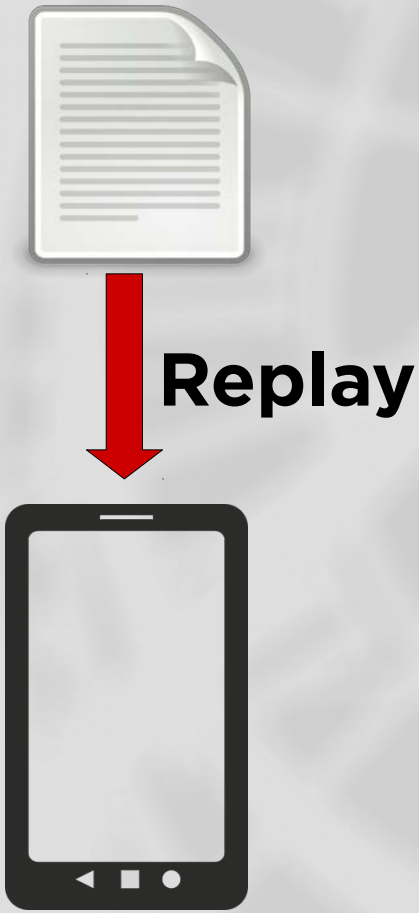
Realistic

Interactive

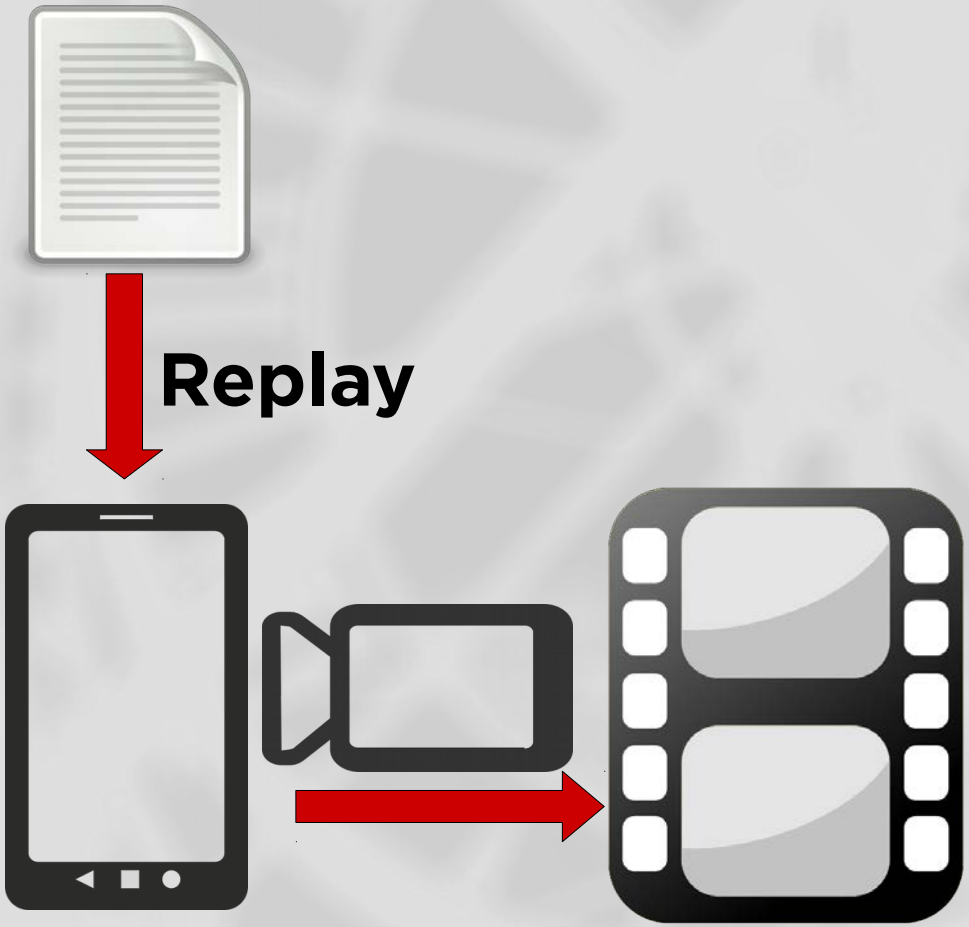
Repeatable

Sample

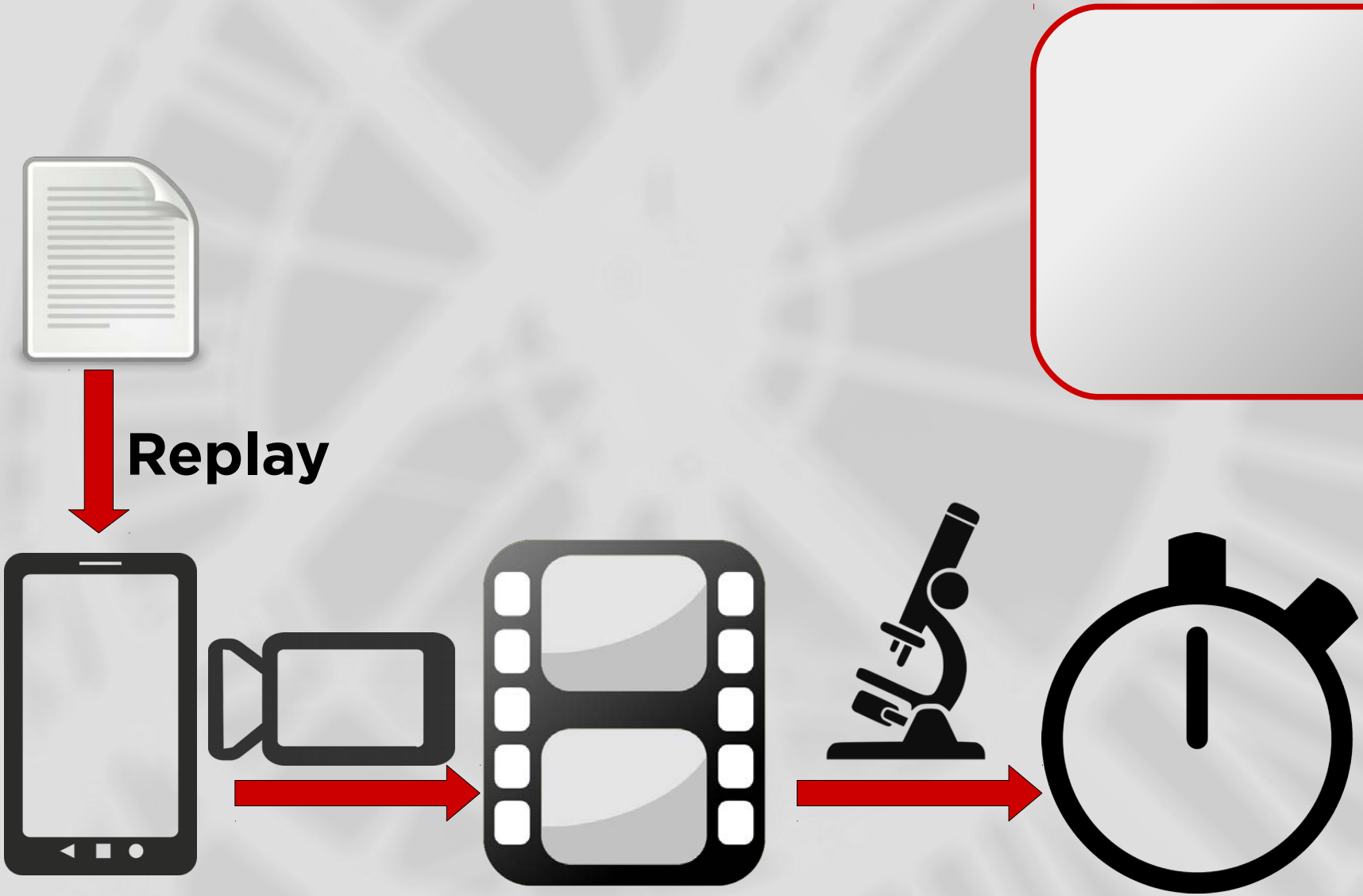
Markup



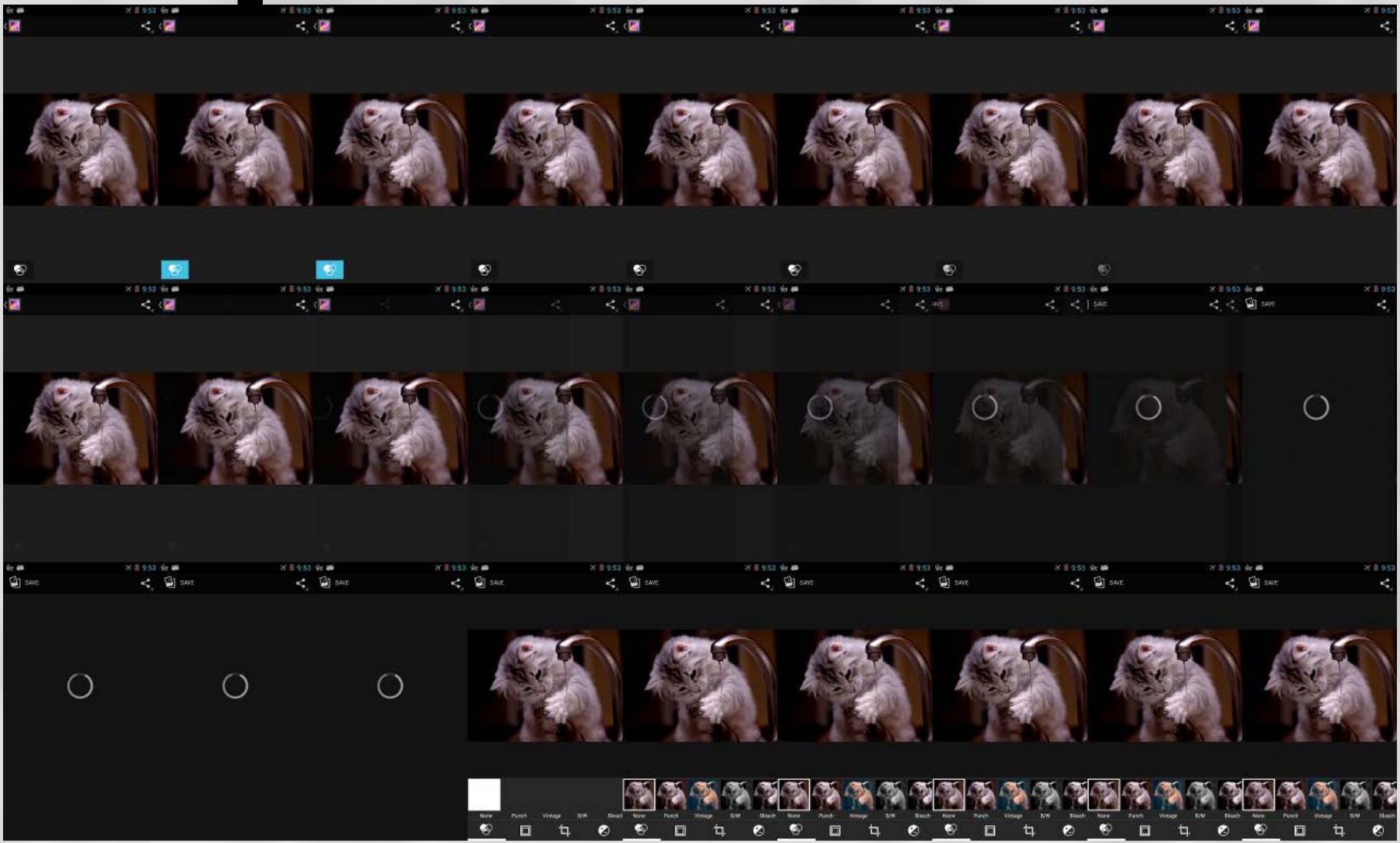
Markup



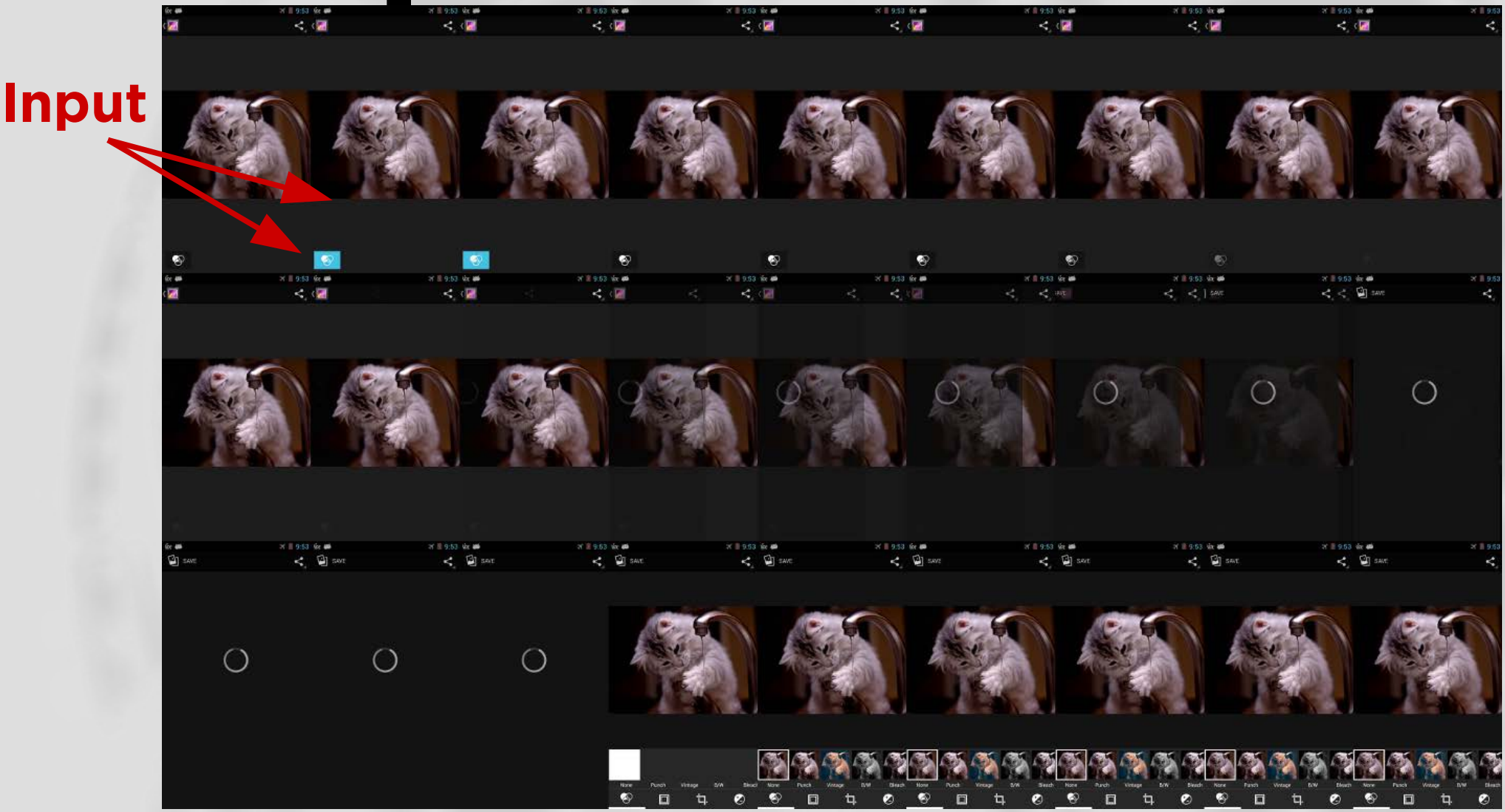
Markup



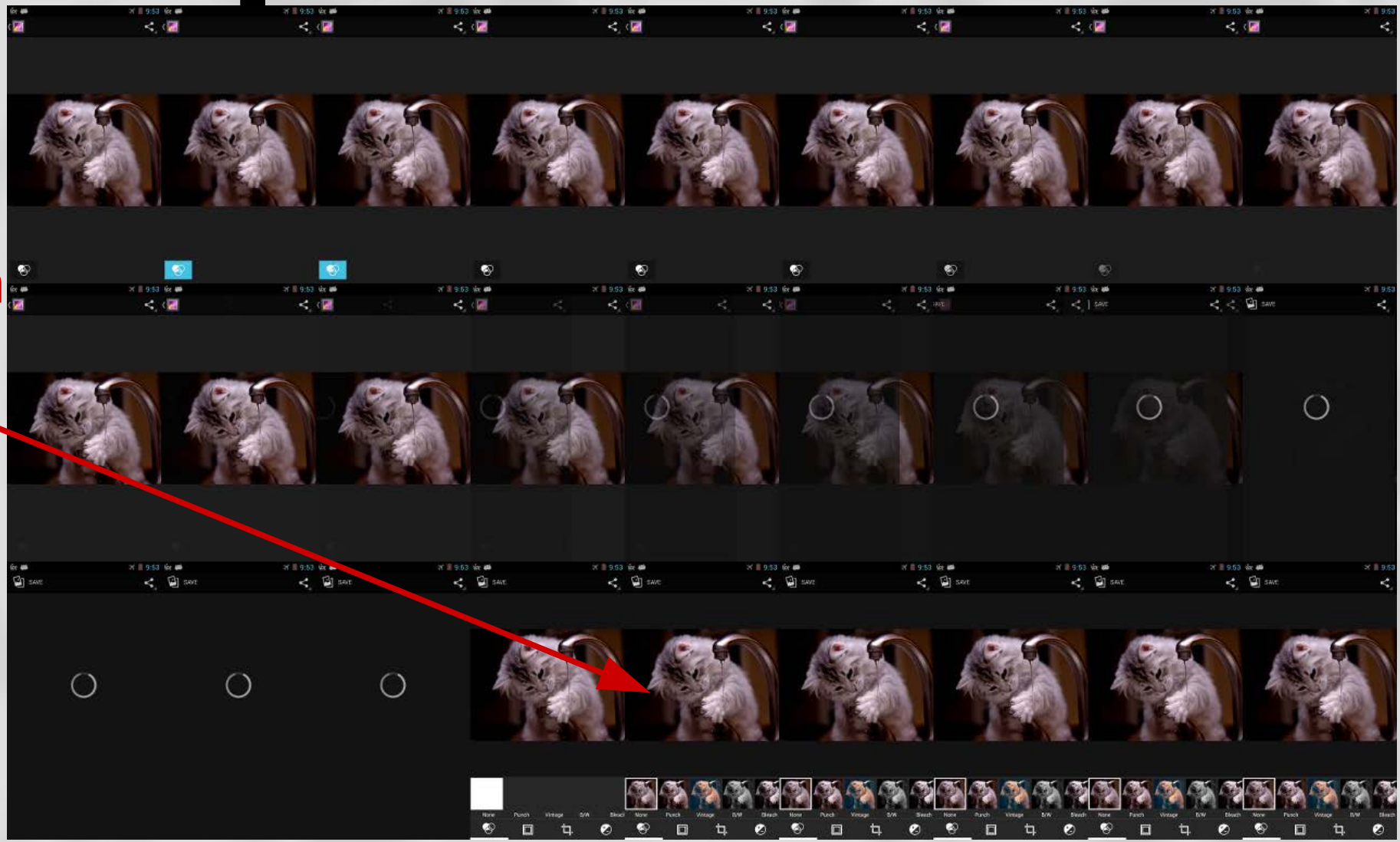
Markup



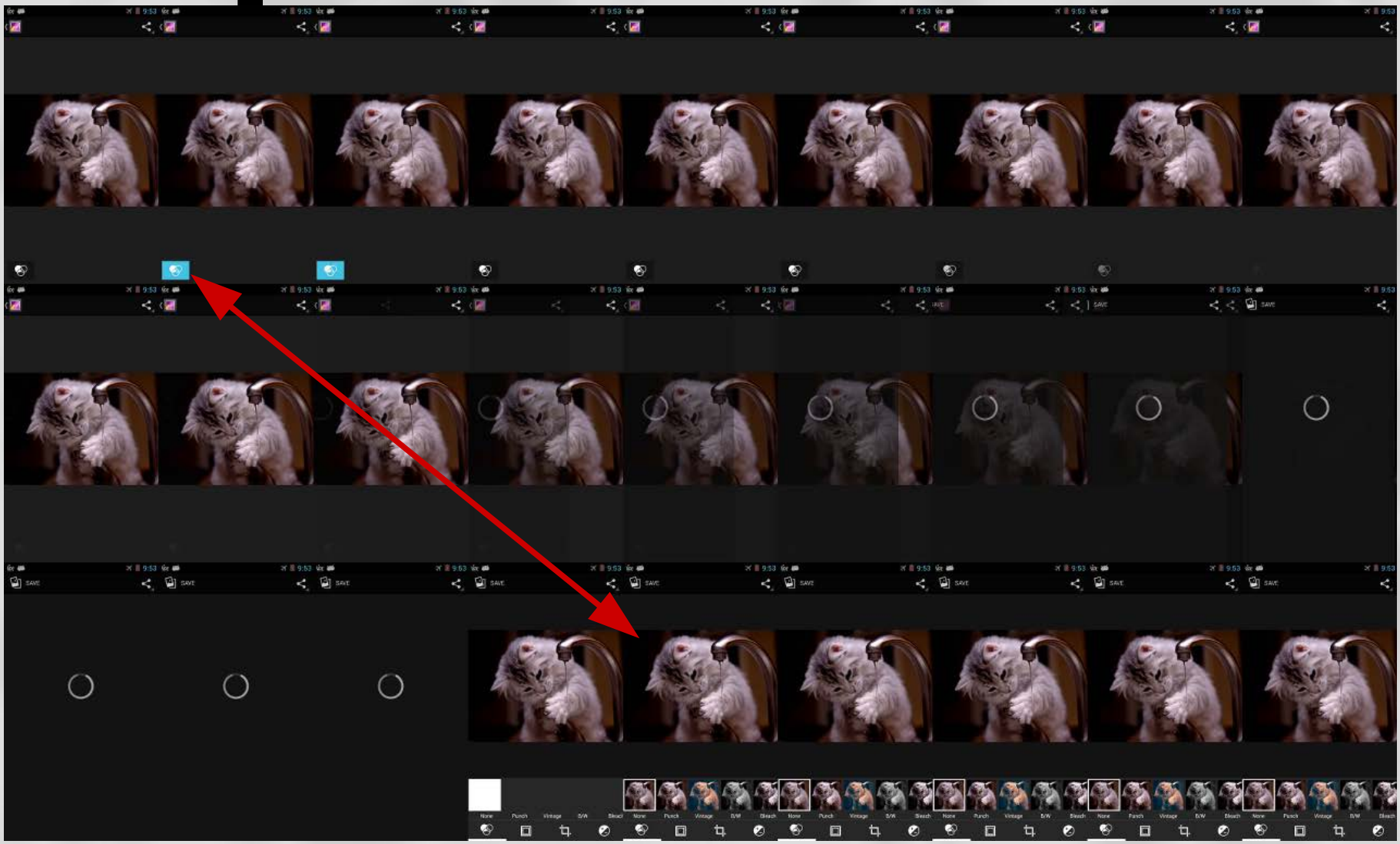
Markup



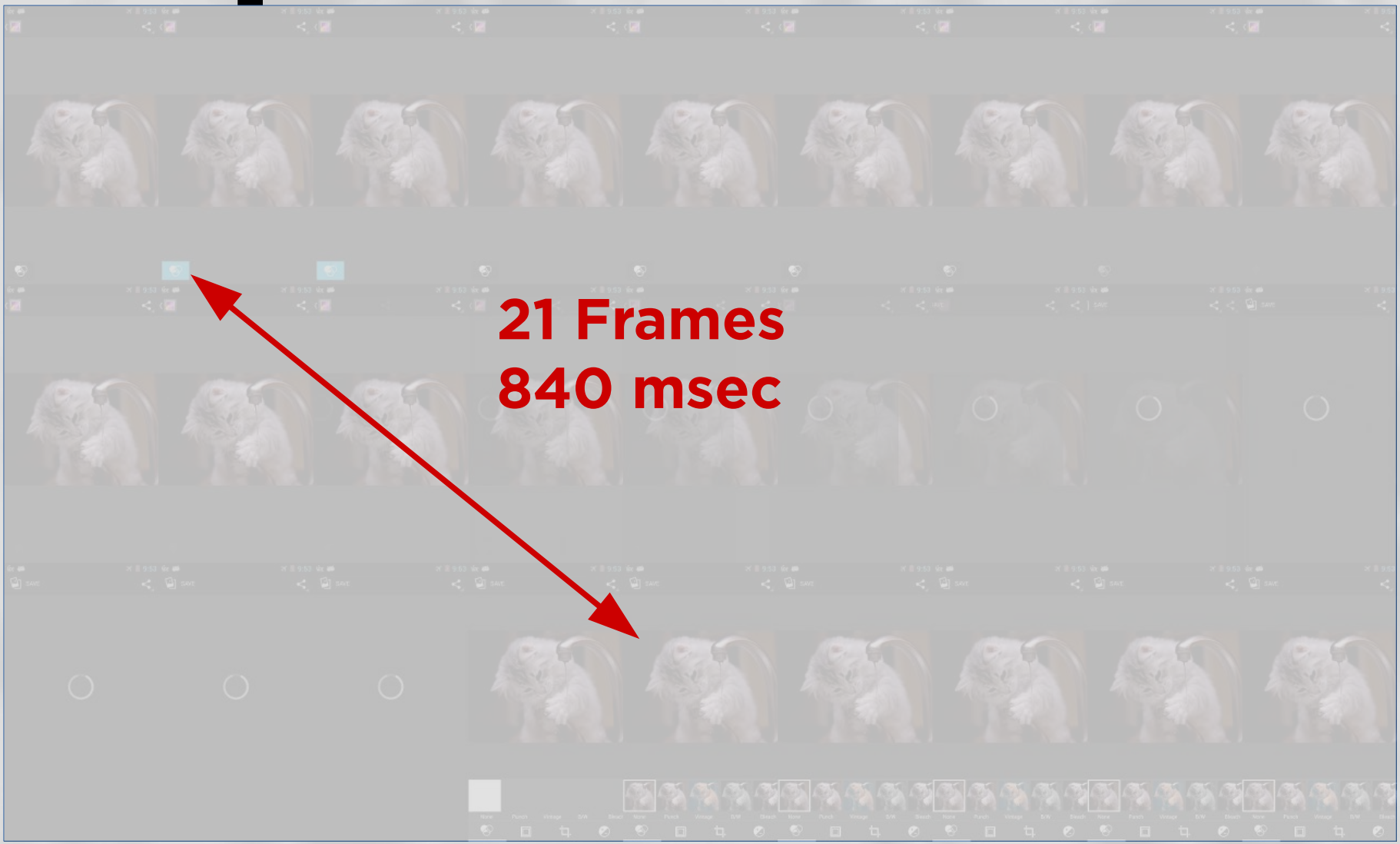
Markup



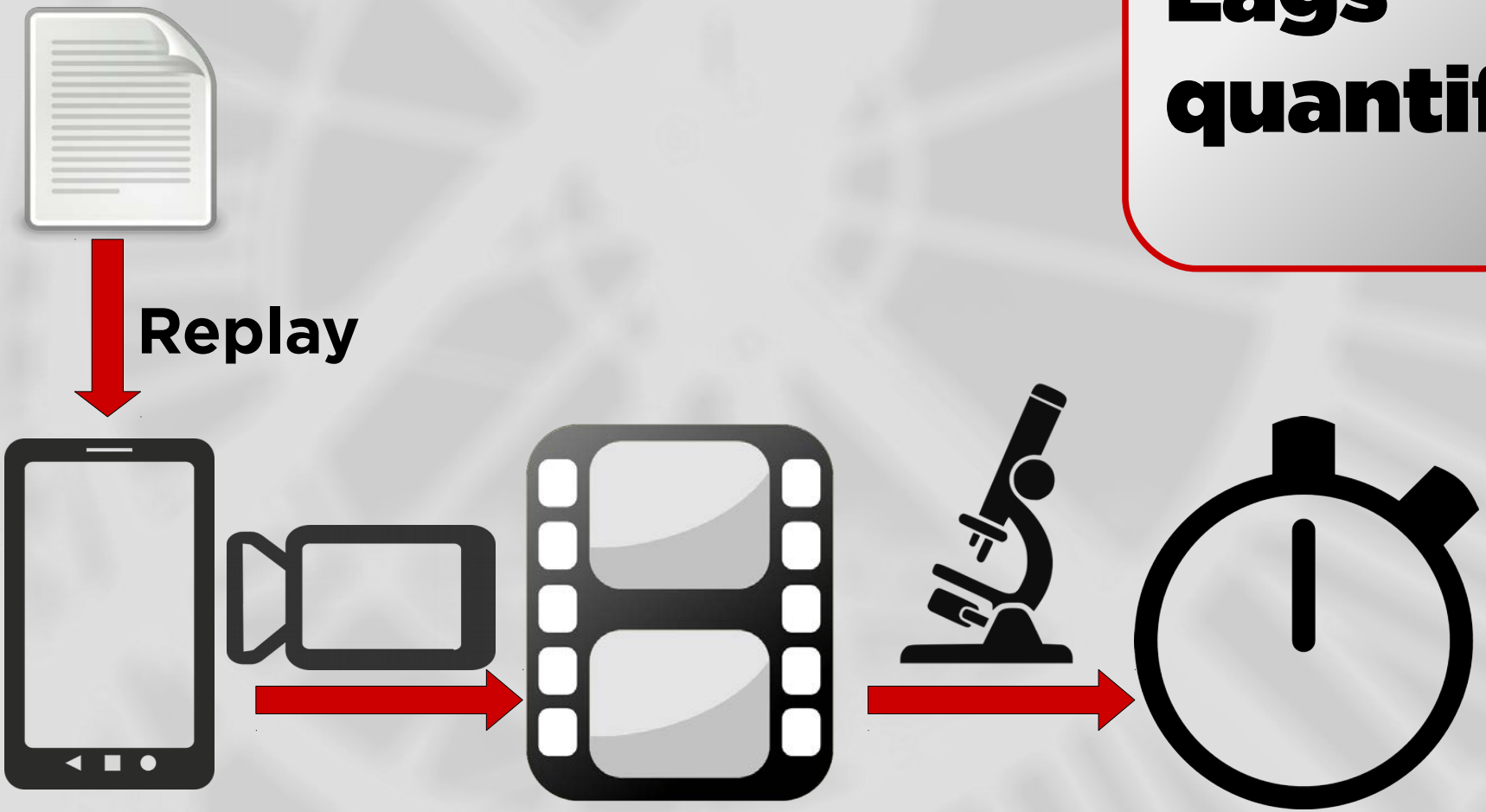
Markup



Markup



Markup

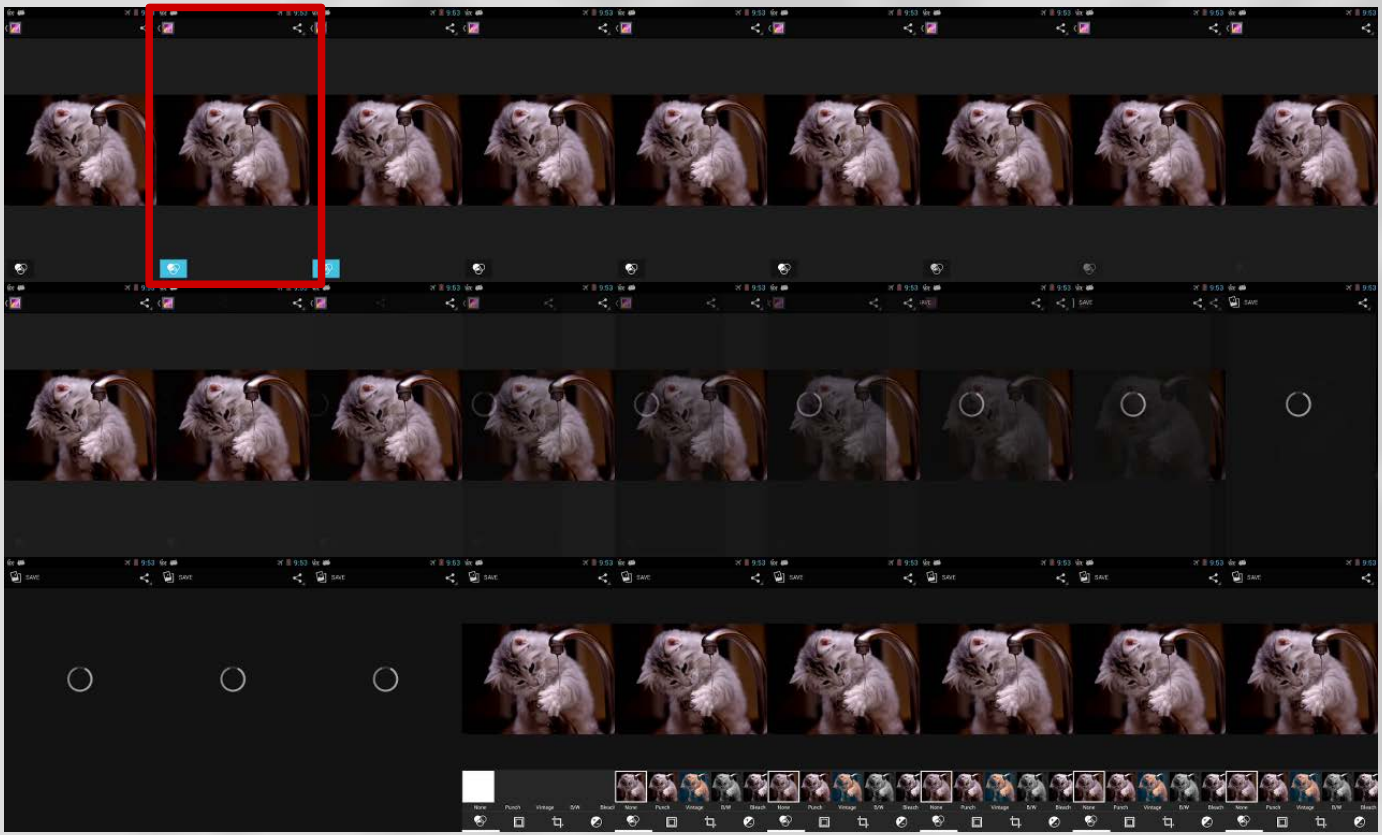


Lags quantified

Markup

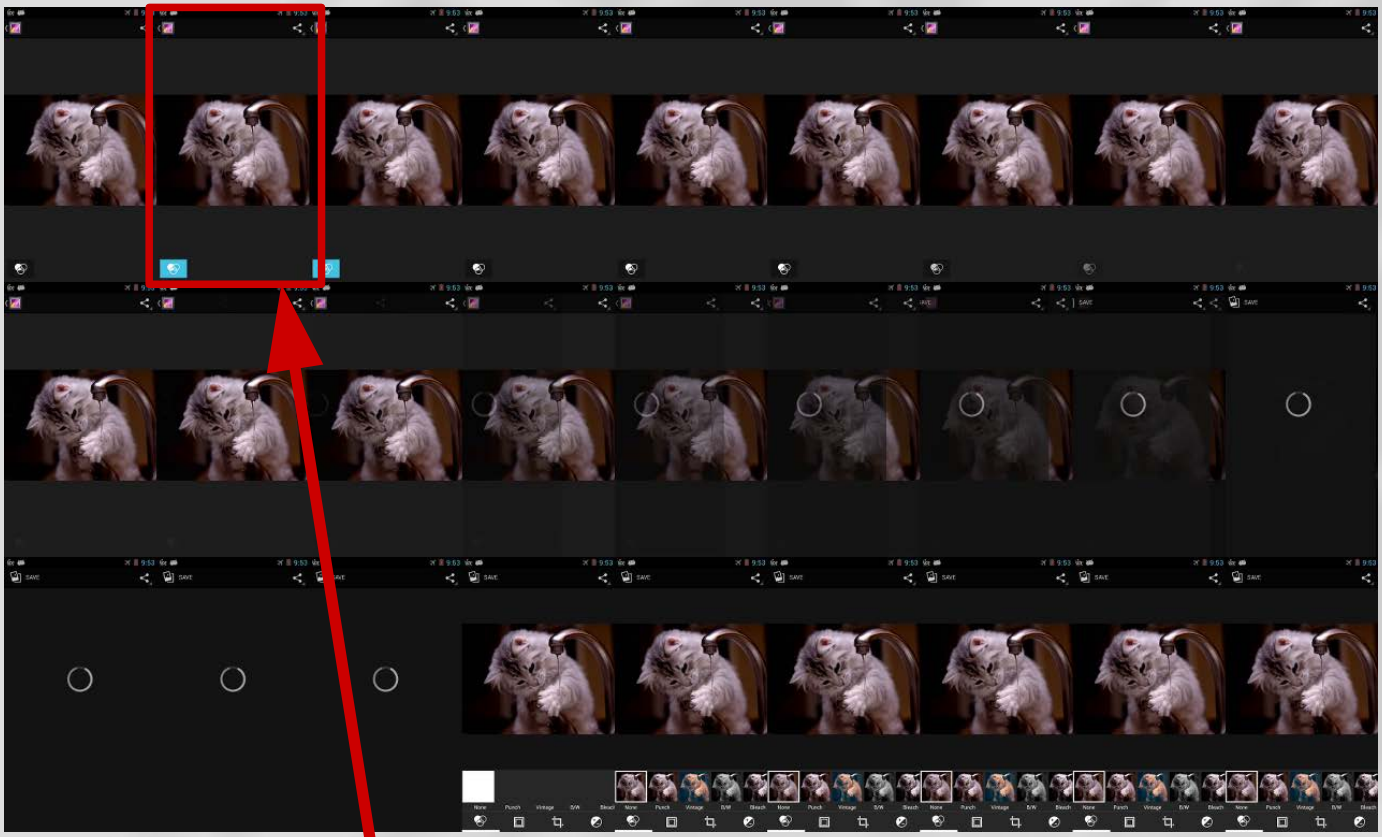
45 sec per lag

Semi-automatic markup



45 sec per lag

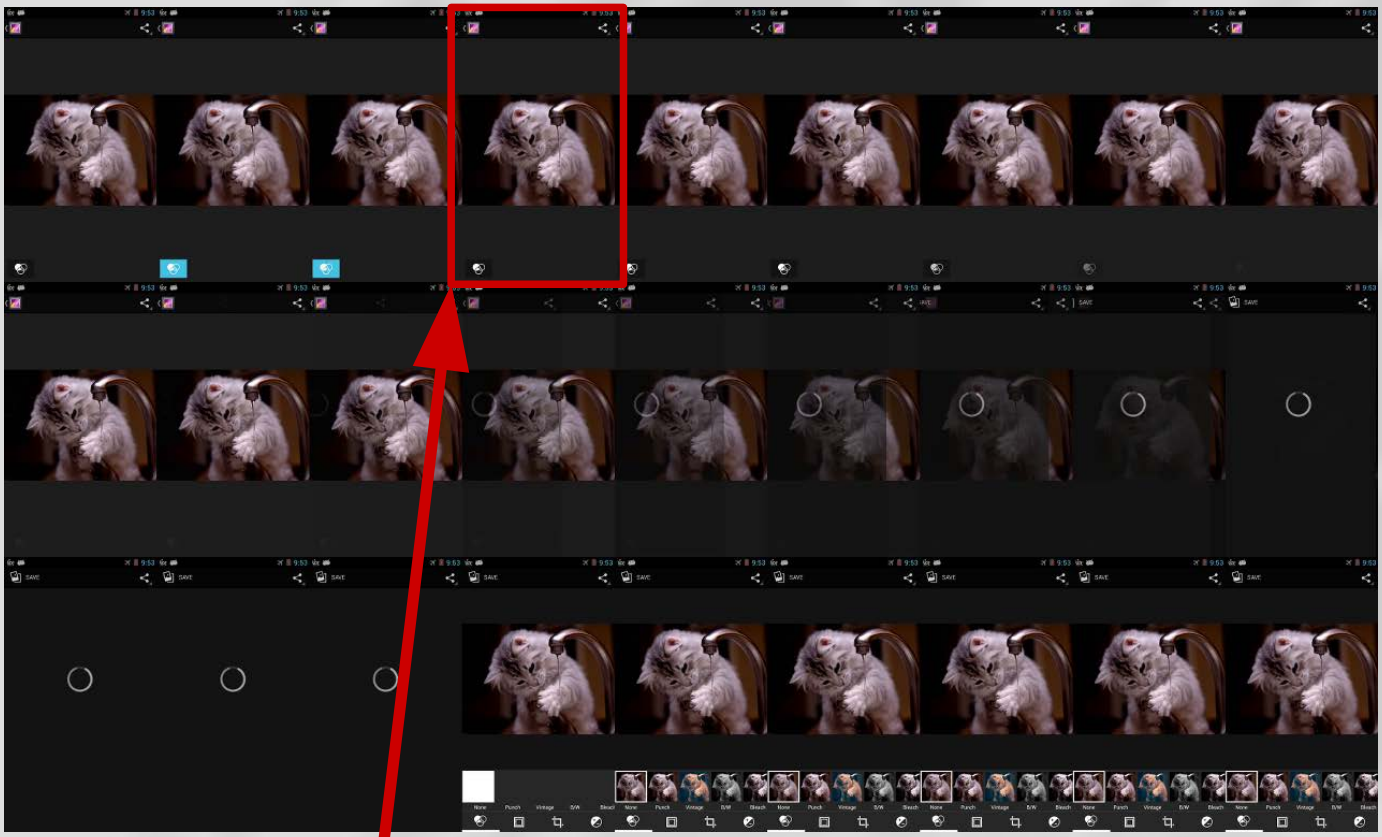
Semi-automatic markup



45 sec per lag

Interaction end after first frame

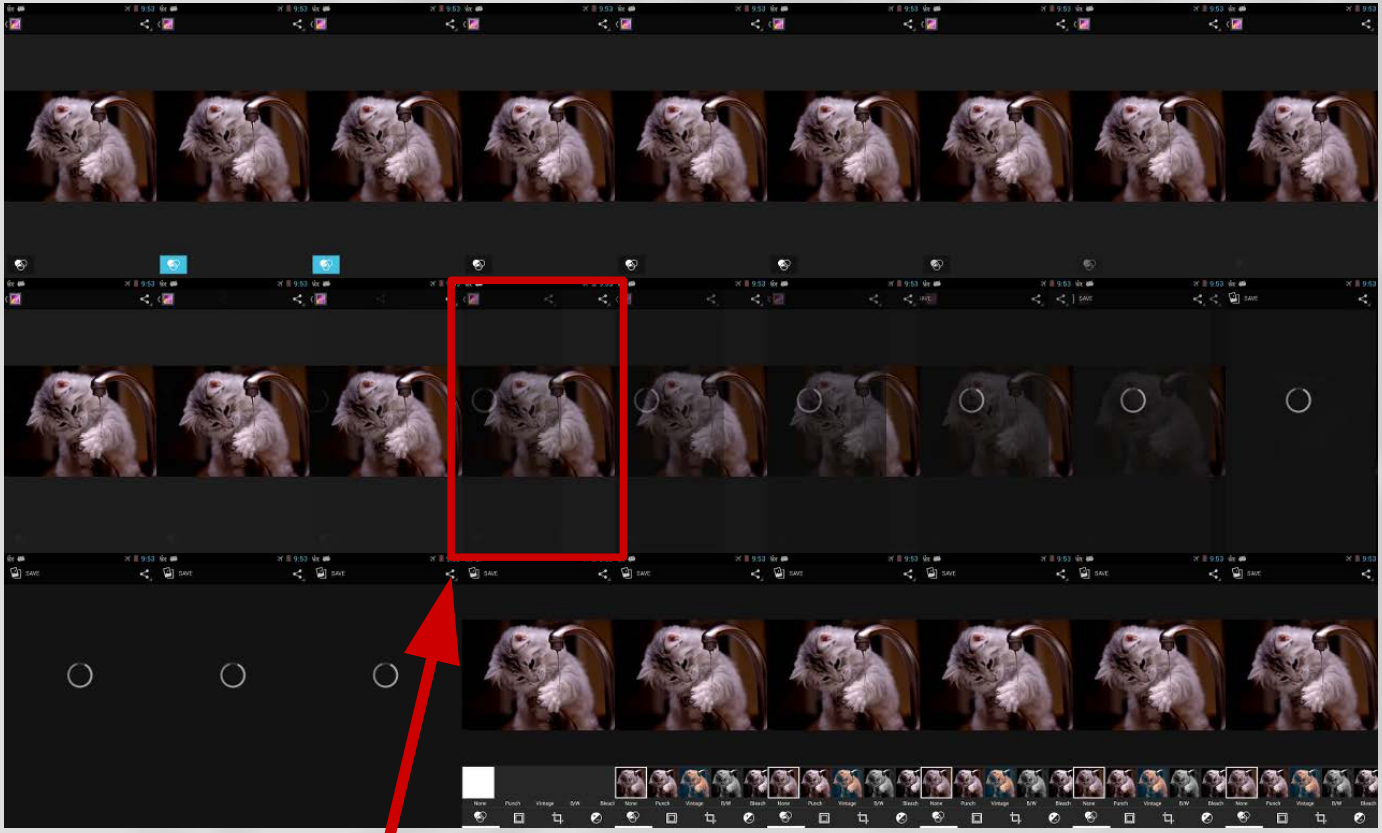
Semi-automatic markup



First Screen Change

45 sec per lag

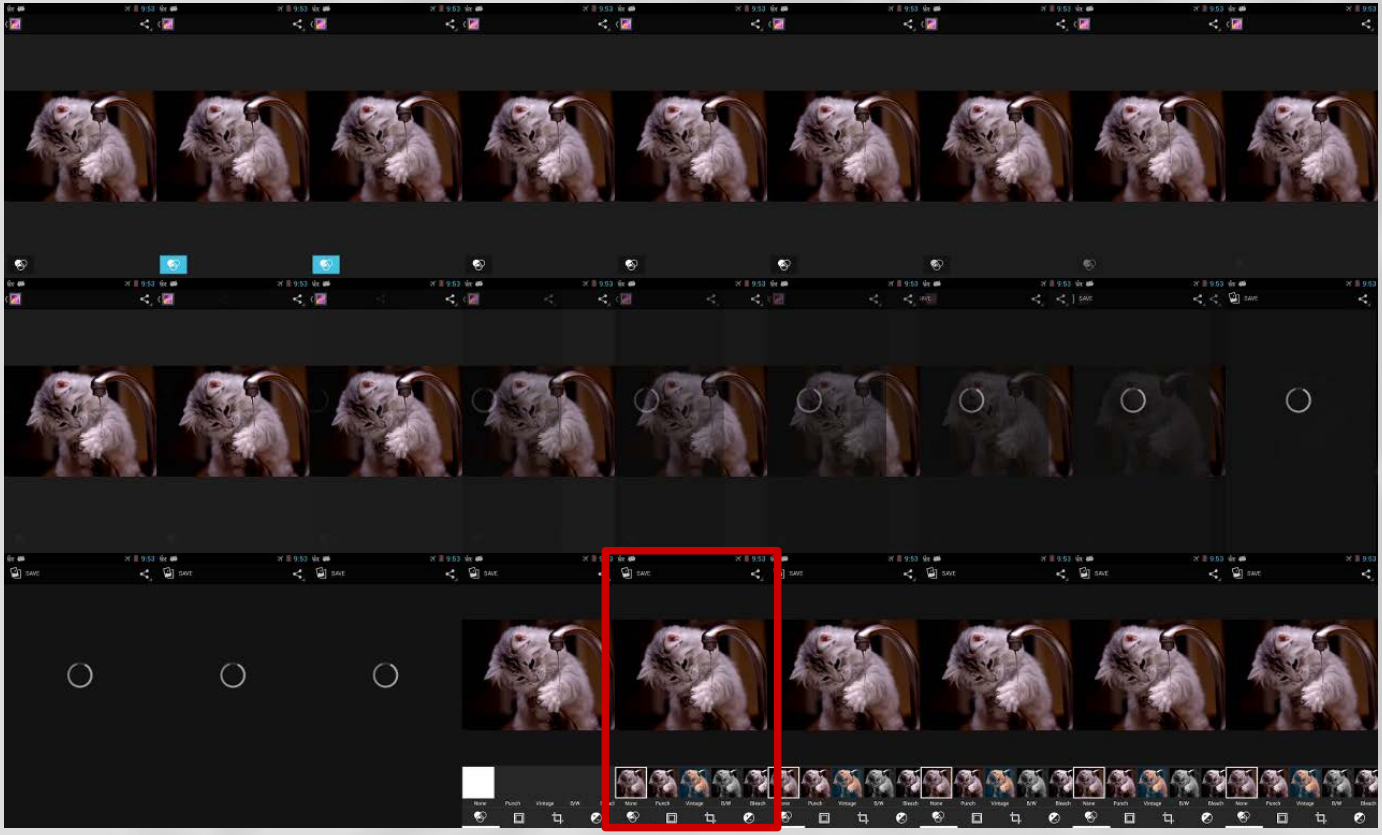
Semi-automatic markup



45 sec per lag

Screen Changes after frames of no change

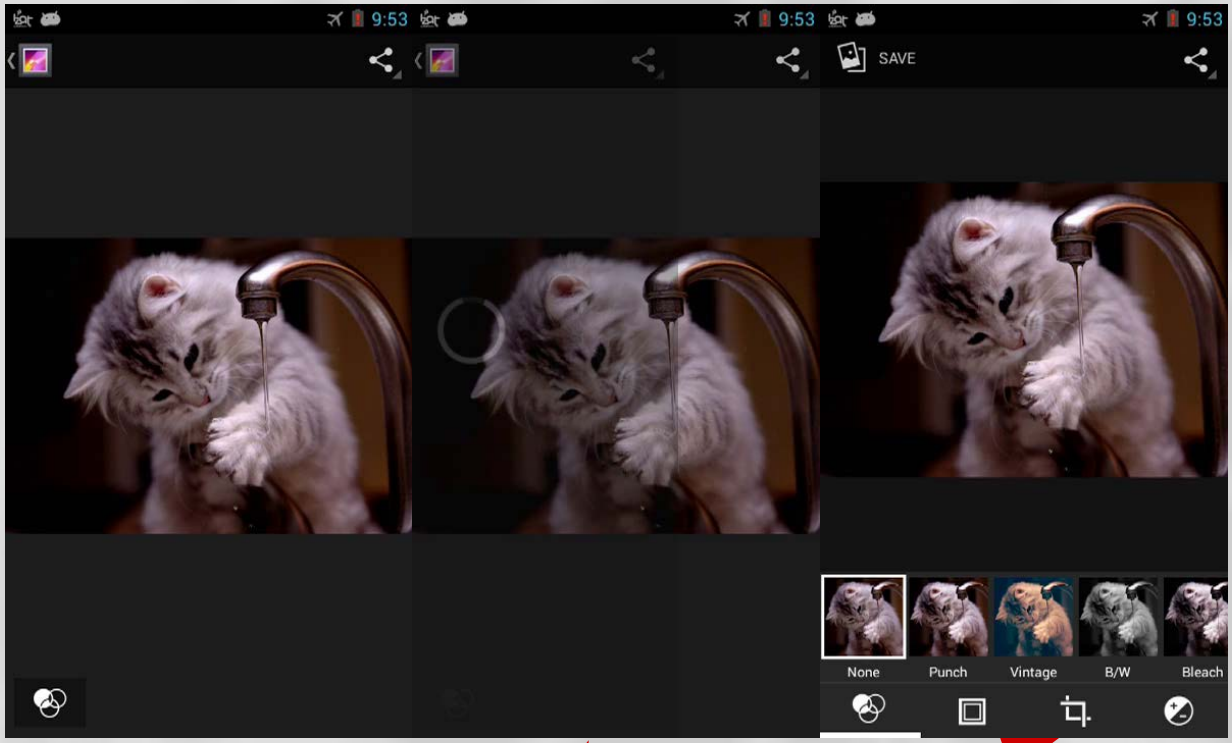
Semi-automatic markup



45 sec per lag

Screen stops changing

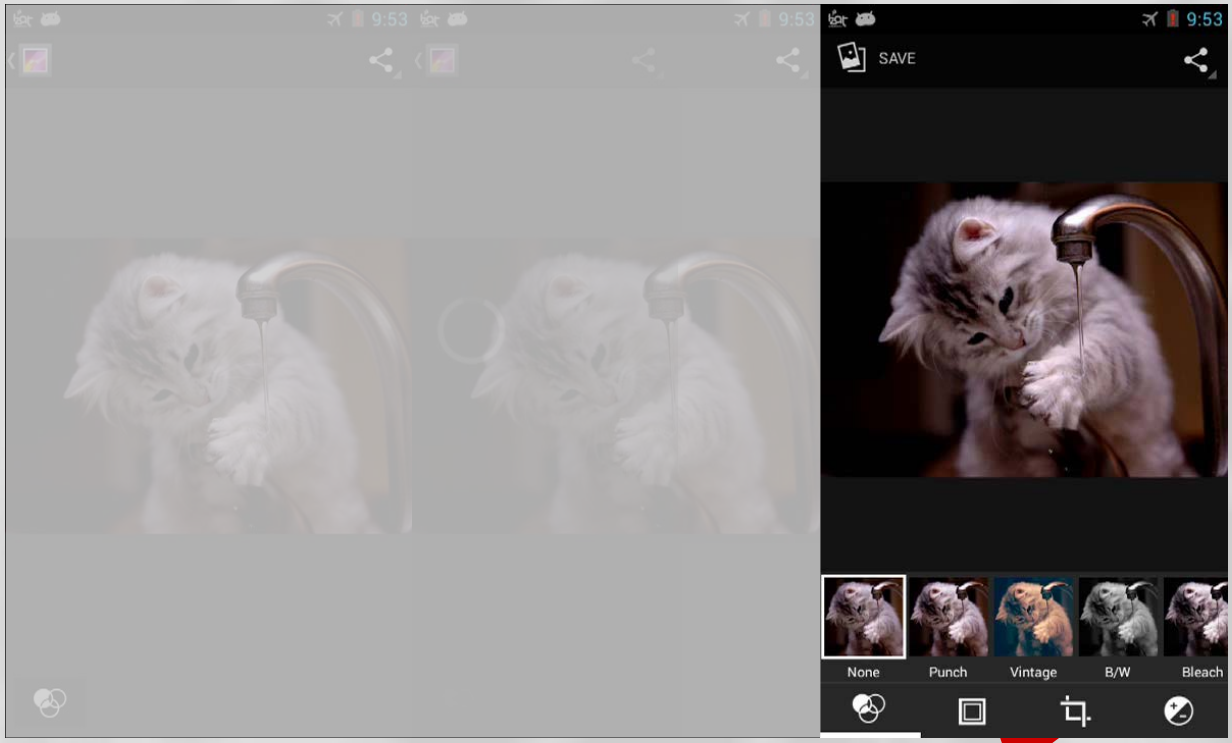
Semi-automatic markup



3 frames to choose from

45 sec per lag

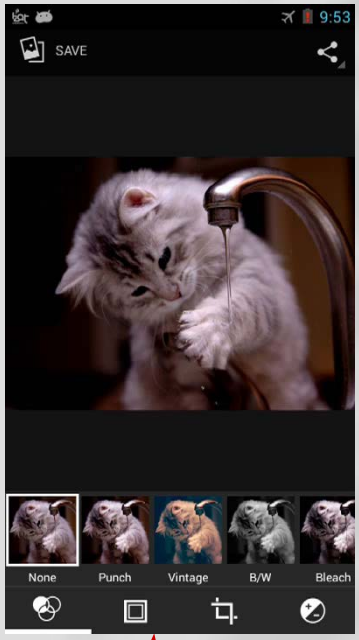
Semi-automatic markup



2-5 sec per lag

Interaction End

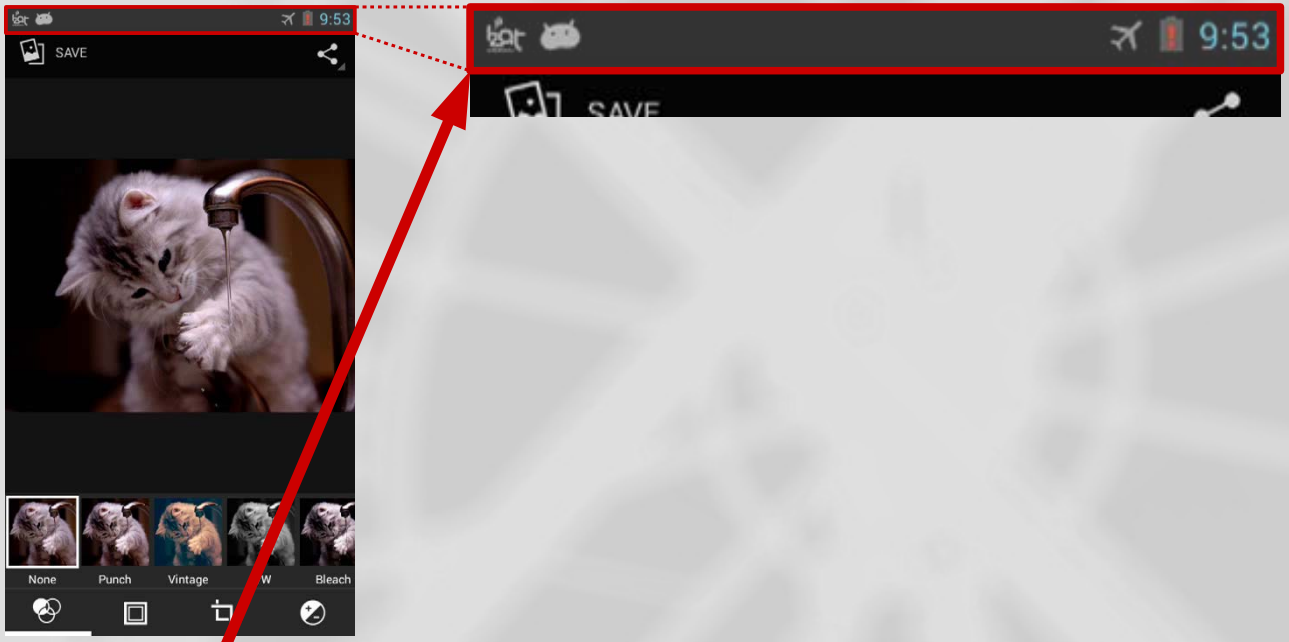
Semi-automatic markup



Save frame

2-5 sec per lag

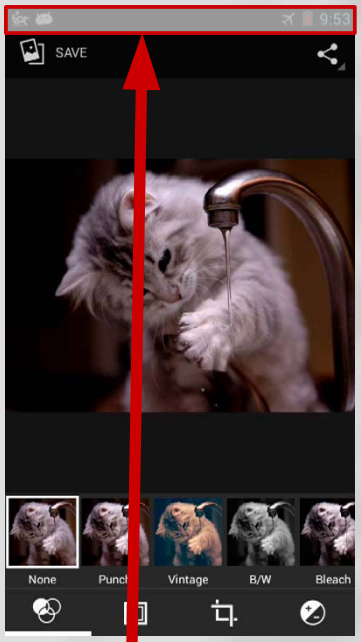
Semi-automatic markup



Variable area

2-5 sec per lag

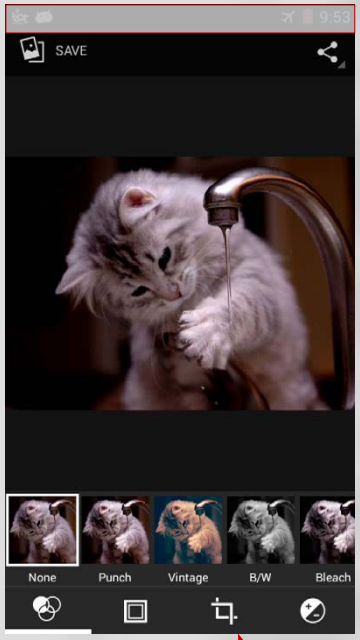
Semi-automatic markup

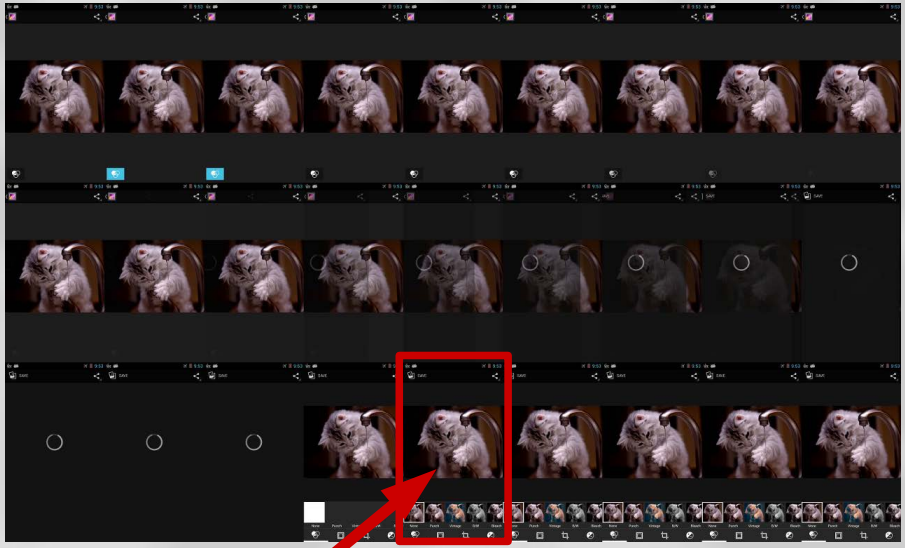


Masked area

2-5 sec per lag

Semi-automatic markup





Human input needed only once

Interaction End

Representative

Representative

Real Mobile Applications

Real Inputs

Real Metrics

Repeatable

Repeatable

Same behaviour every time

Automatic

Automatic

No code analysis

No instrumentation

No humans needed*

*after initial video markup

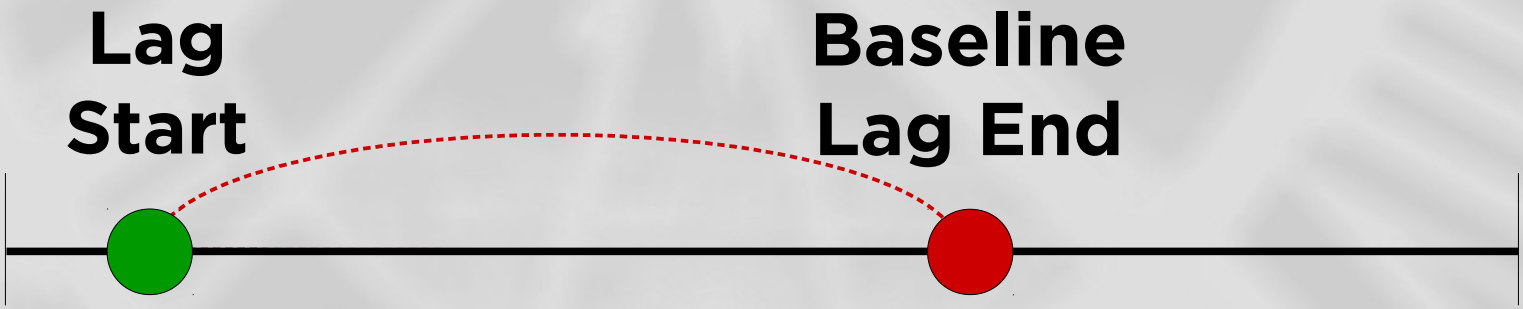
Android Frequency Governors

Android Frequency Governors

Energy VS QoE

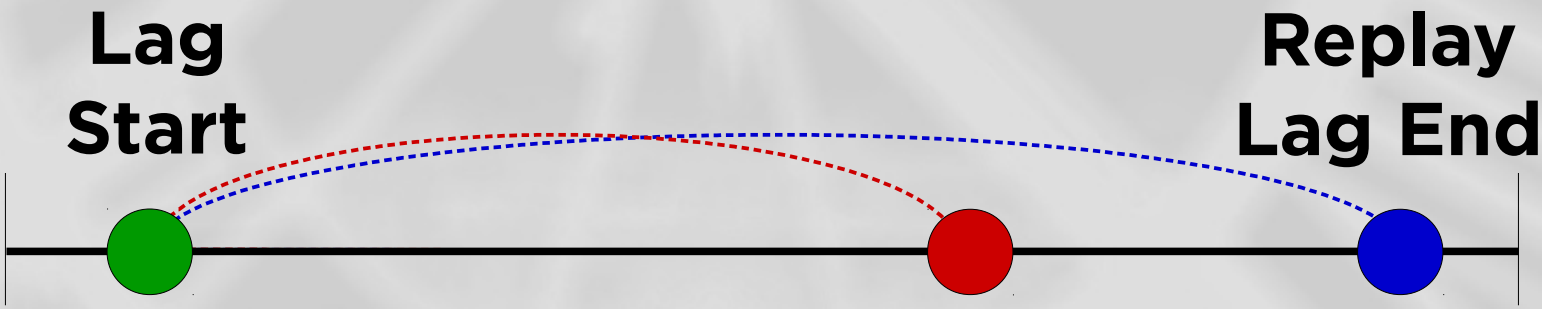
Android Frequency Governors

Energy VS QoE



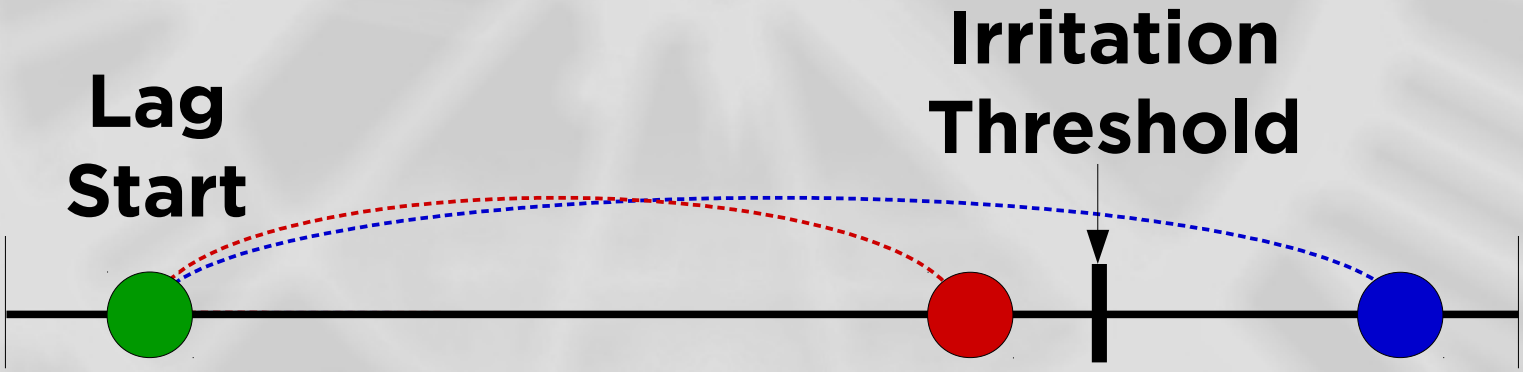
Android Frequency Governors

Energy VS QoE



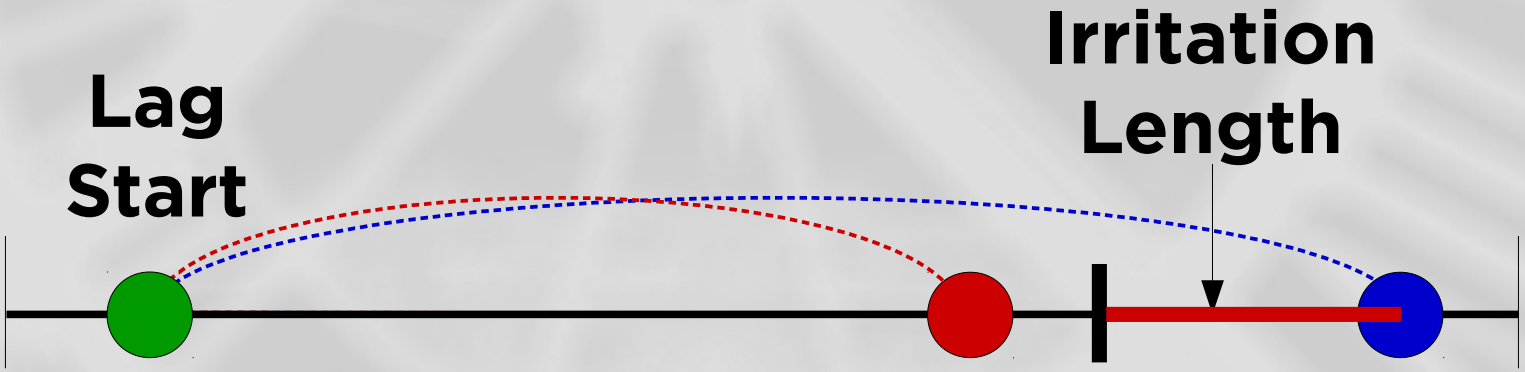
Android Frequency Governors

Energy VS QoE



Android Frequency Governors

Energy VS QoE



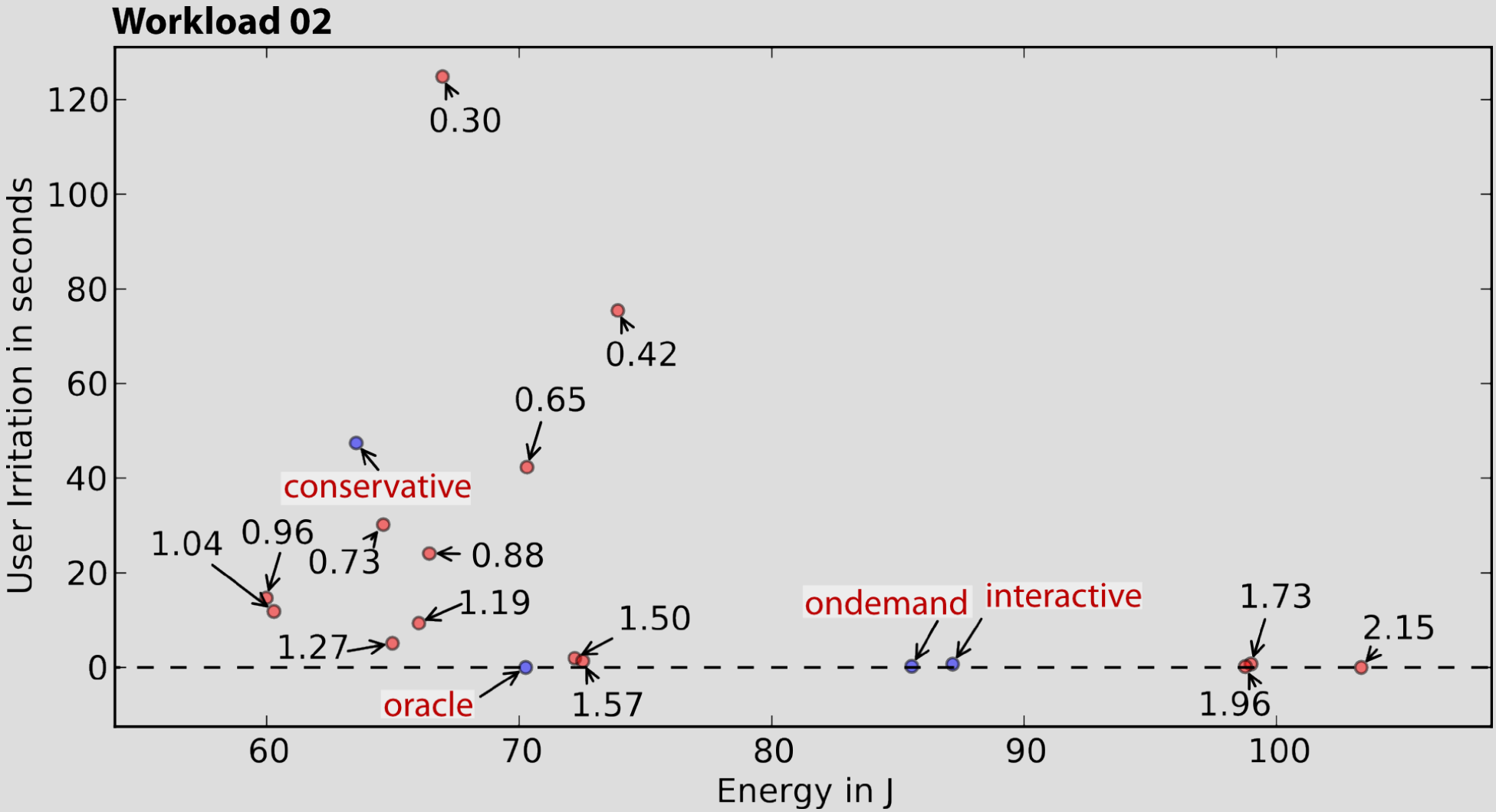
5 workloads

14 different setups

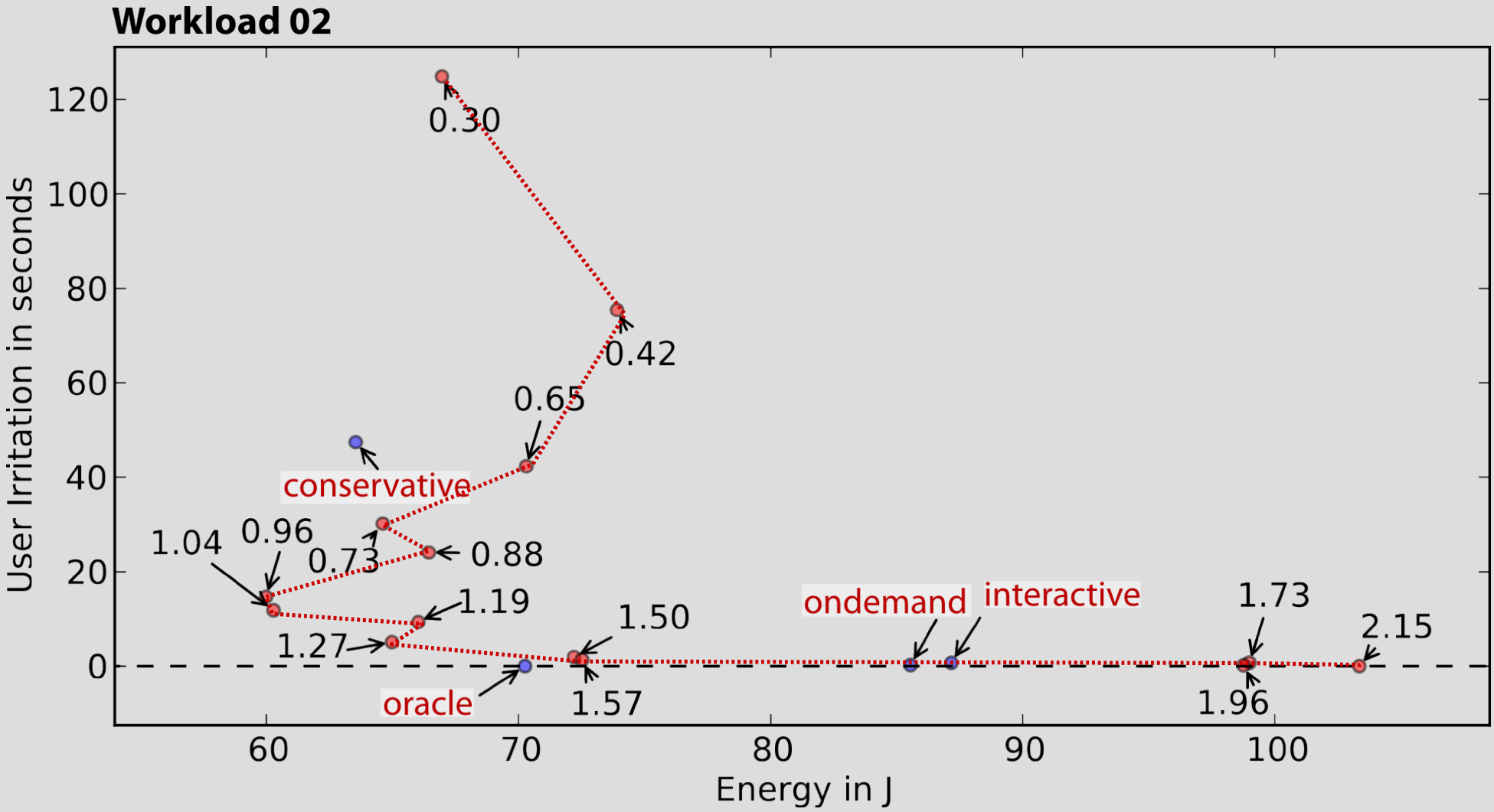
5 repeats each

350 exper. & 2.5 days of video

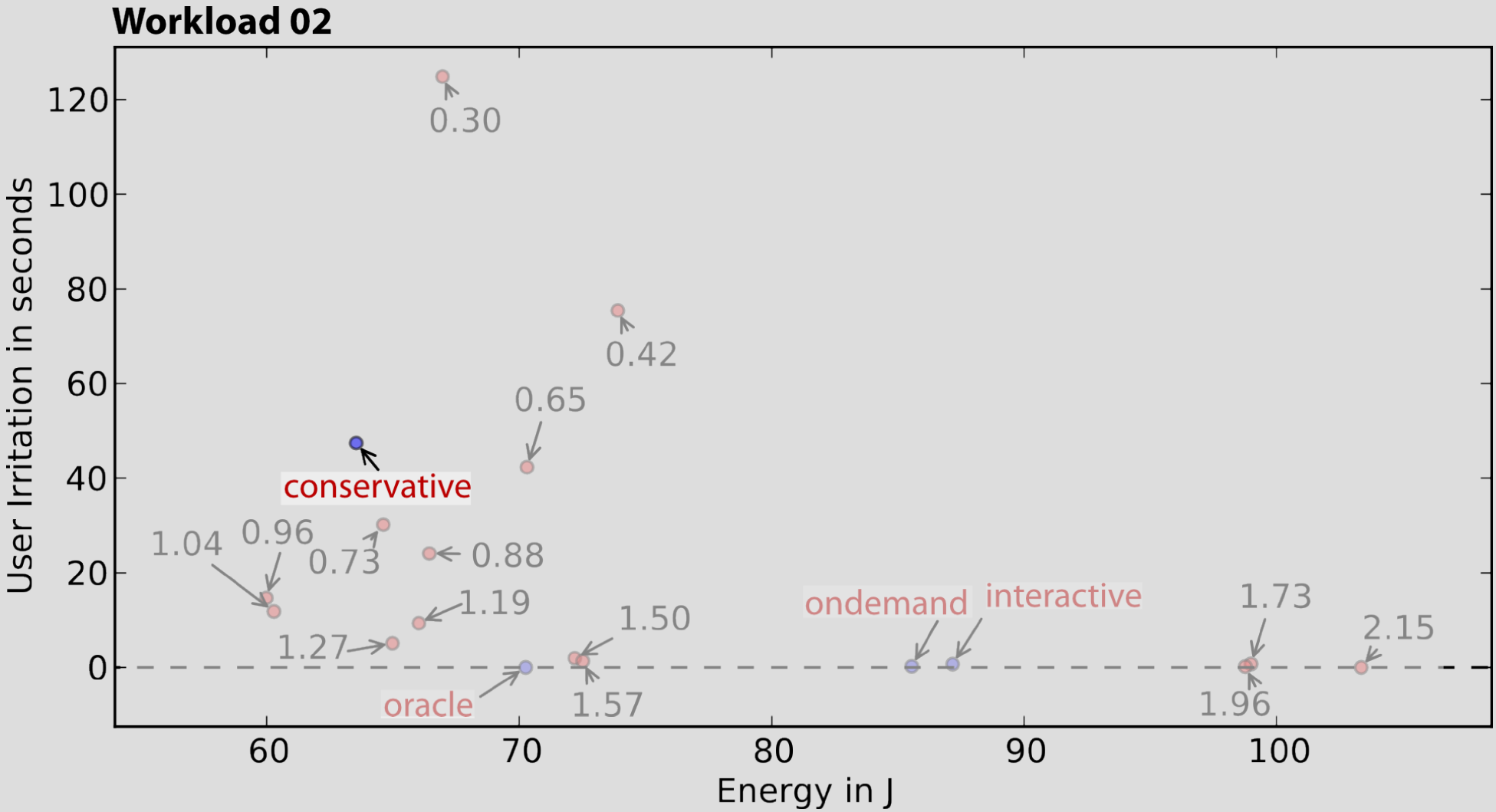
Energy vs QoE



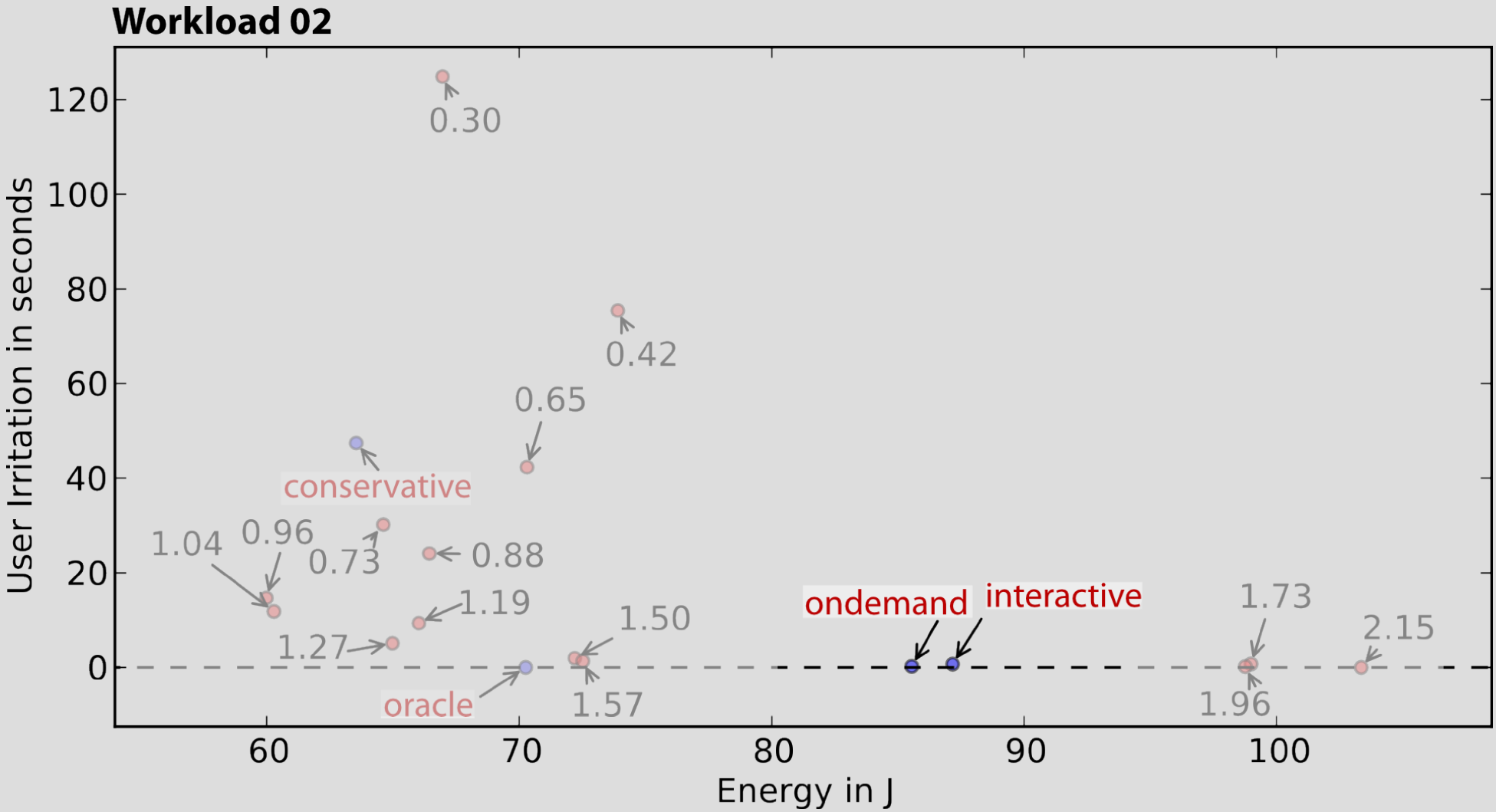
Energy vs QoE



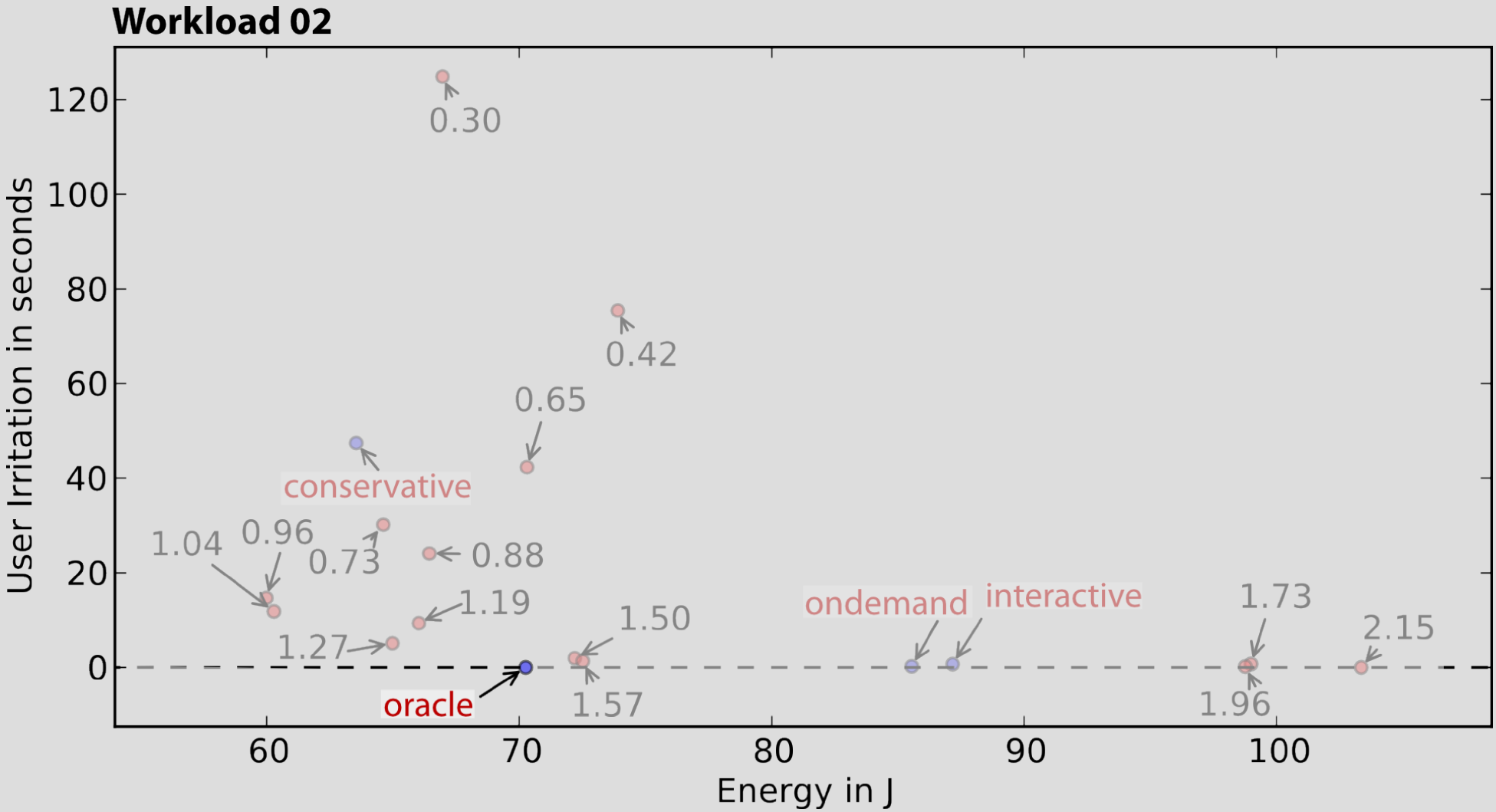
Energy vs QoE



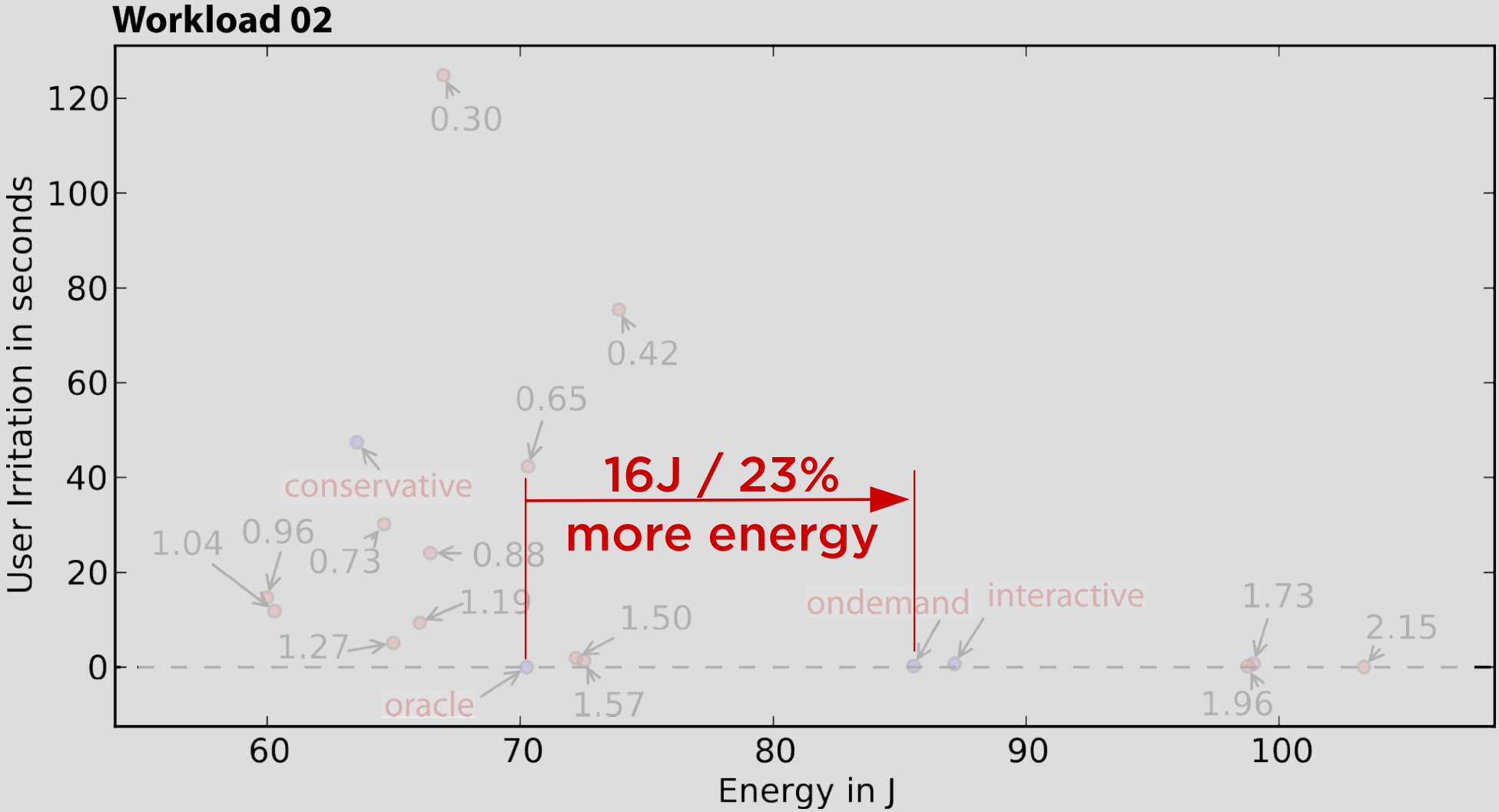
Energy vs QoE



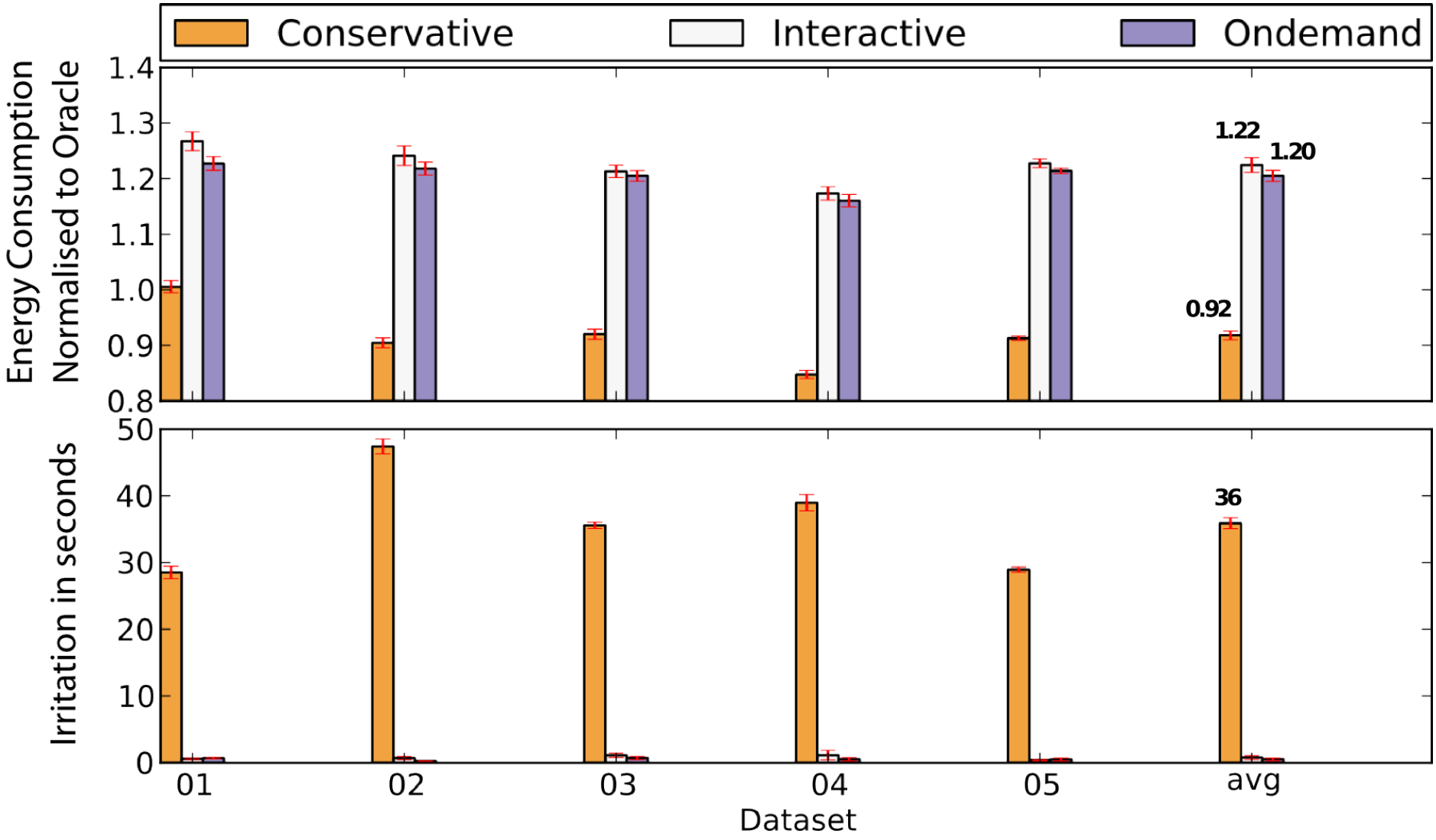
Energy vs QoE



Energy vs QoE



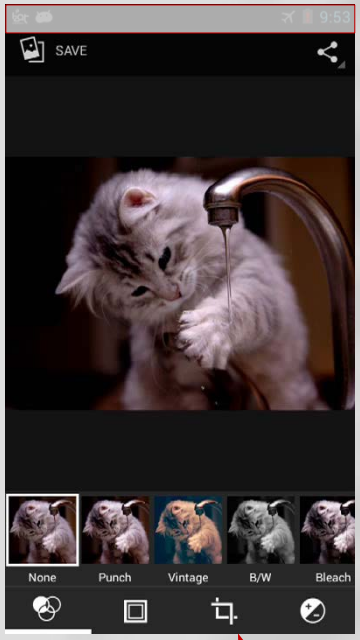
Energy vs QoE

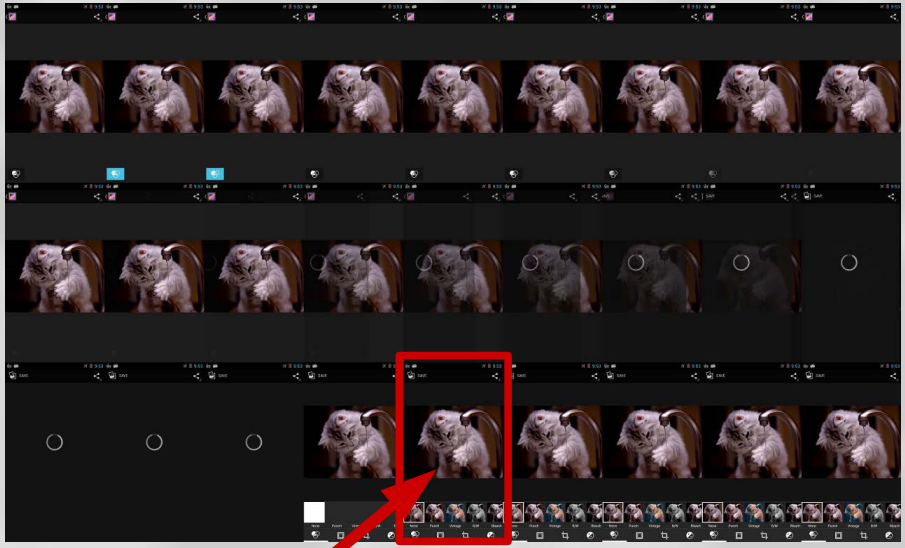


Next Steps

Fully Automatic Markup

Fully automatic markup

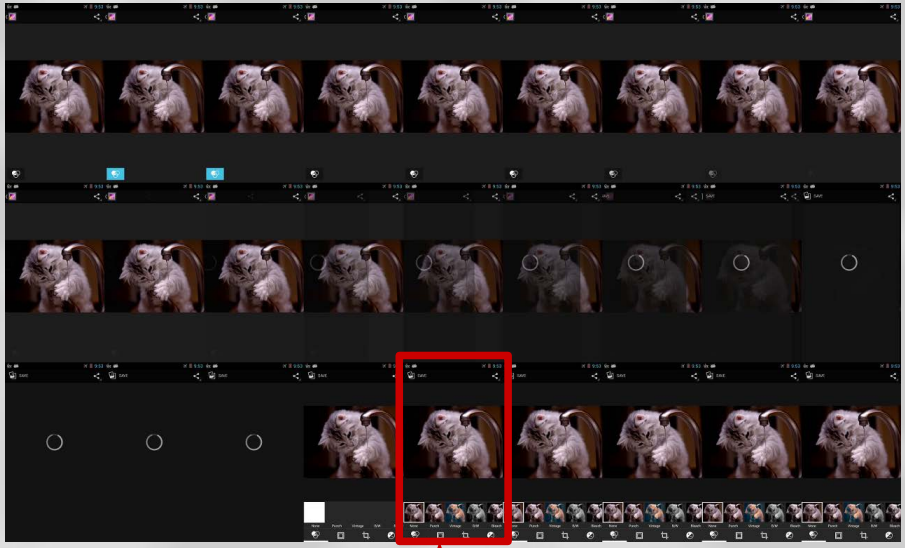




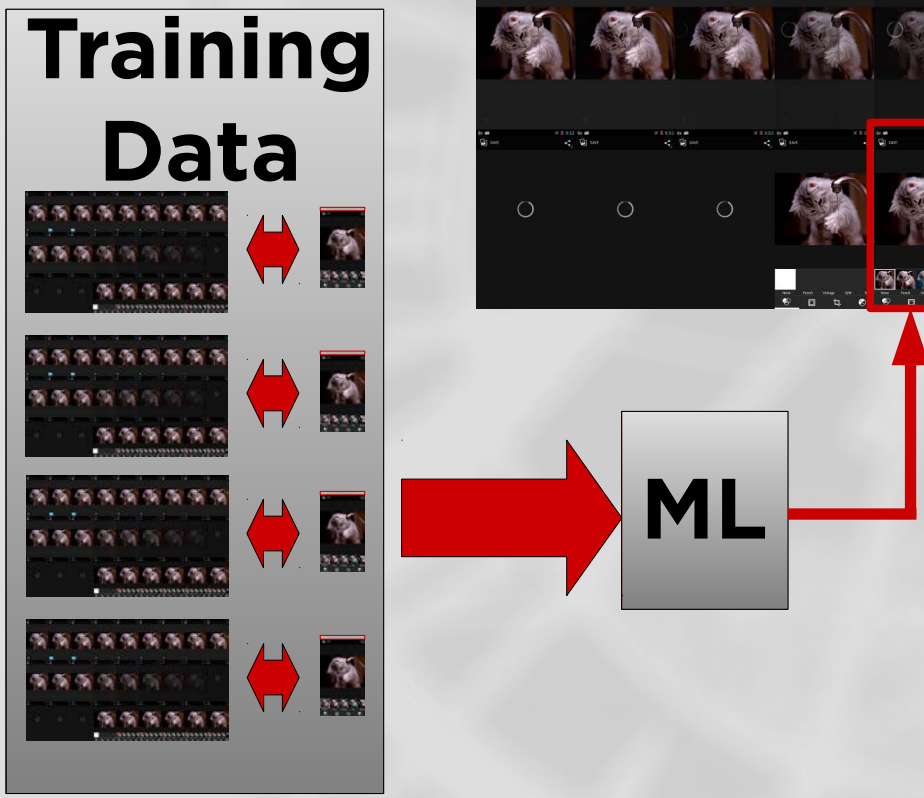
Human input needed only once

Interaction End

Fully automatic markup



Human input needed only for training



After the fact

Lag End Estimation

Can only identify lags offline

Want to identify lags online

Before the fact

Lag End Predictor

Can tell whether the interaction has ended

Want to tell when the interaction will end

QoE for everything

QoE for everything

Can test/fine-tune heuristics offline

QoE for everything

Can test/fine-tune heuristics offline

Will evaluate them online

QoE for everything

Can test/fine-tune heuristics offline

Will evaluate them online

Will adapt them online

QoE for everything

Can test/fine-tune heuristics offline

Will evaluate them online

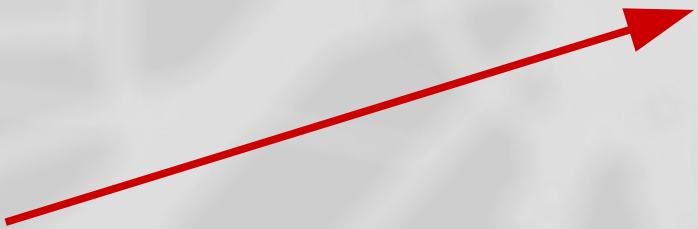
Will adapt them online

For what users care about

Personalised fast optimisations

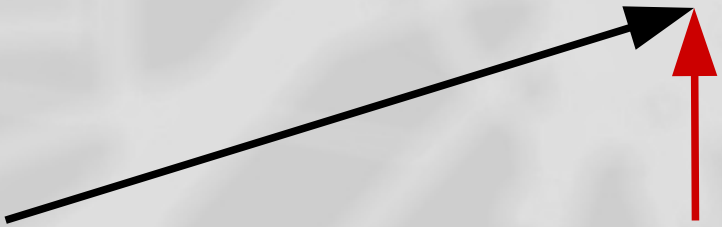
Optimising your application is great!

Optimising your application is great!



Choose your device

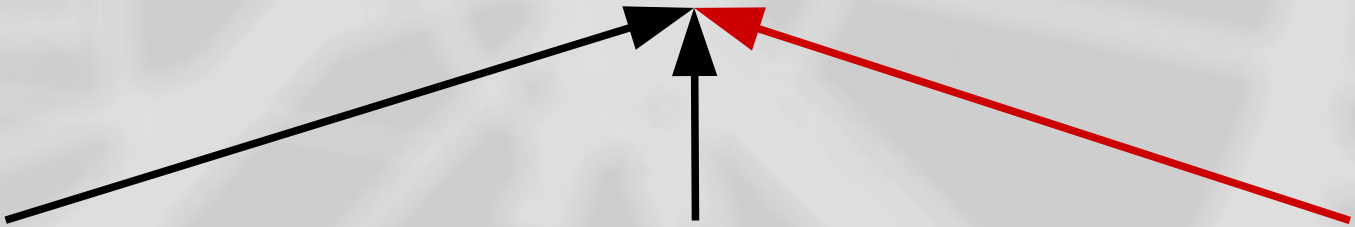
Optimising your application is great!



Choose your device

Choose your compiler flags

Optimising your application is great!



Choose your device

Choose your compiler flags

Choose your runtime parameters

Great but not always easy

Mobile Apps

Data-centre Apps

Mobile Apps

Data-centre Apps

Developers cannot test on every platform

Mobile Apps

Data-centre Apps

Developers cannot test on every platform

System owners don't "understand" the app

Mobile Apps

Data-centre Apps

Developers cannot test on every platform

System owners don't "understand" the app

App may run for hours to years

Mobile Apps

Data-centre Apps

Developers cannot test on every platform

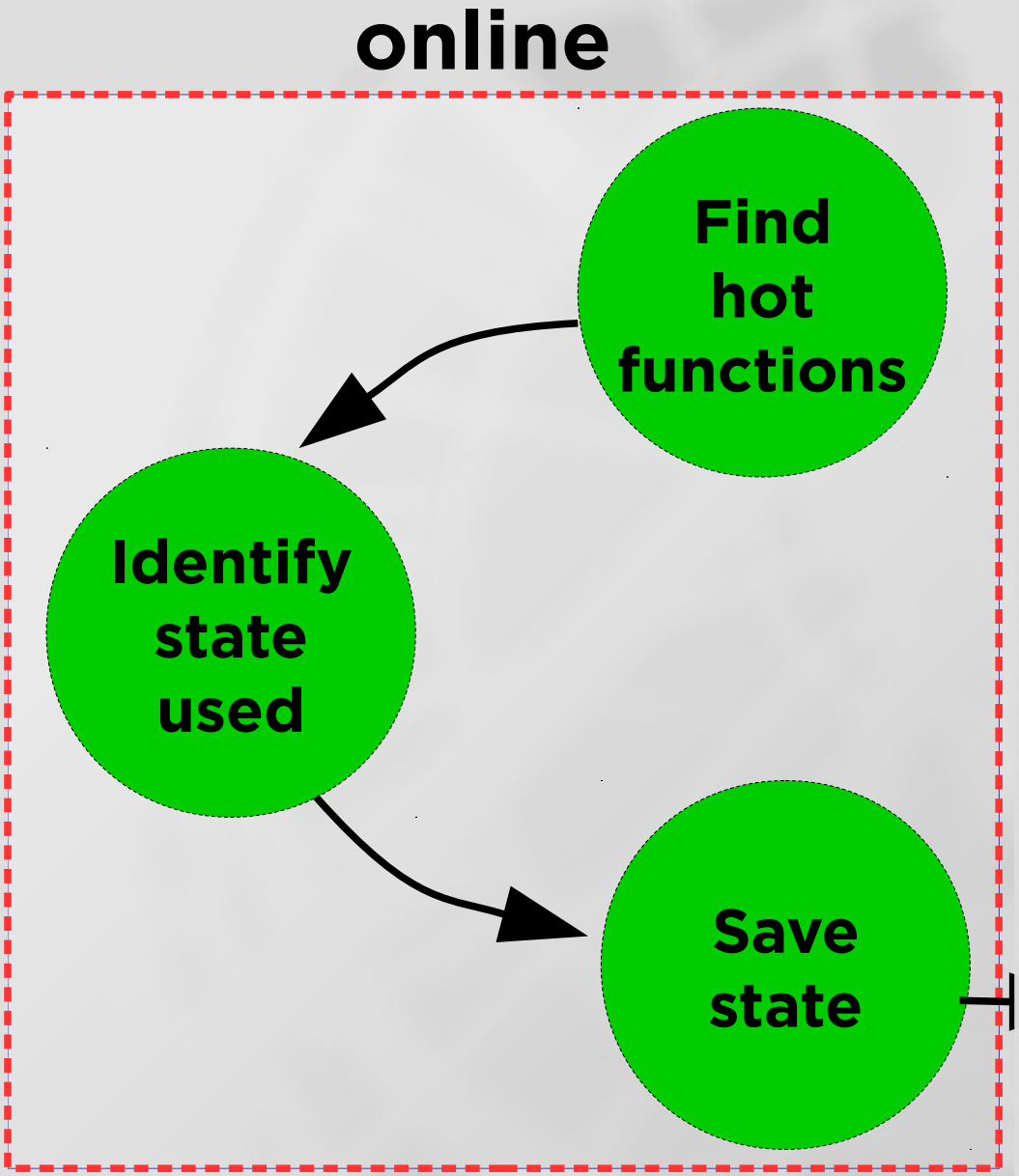
System owners don't "understand" the app

App may run for hours to years

Cannot evaluate optimisations online

How do we optimise in such cases?

Capture & Replay based Optimisation



Low time/space Overhead capture

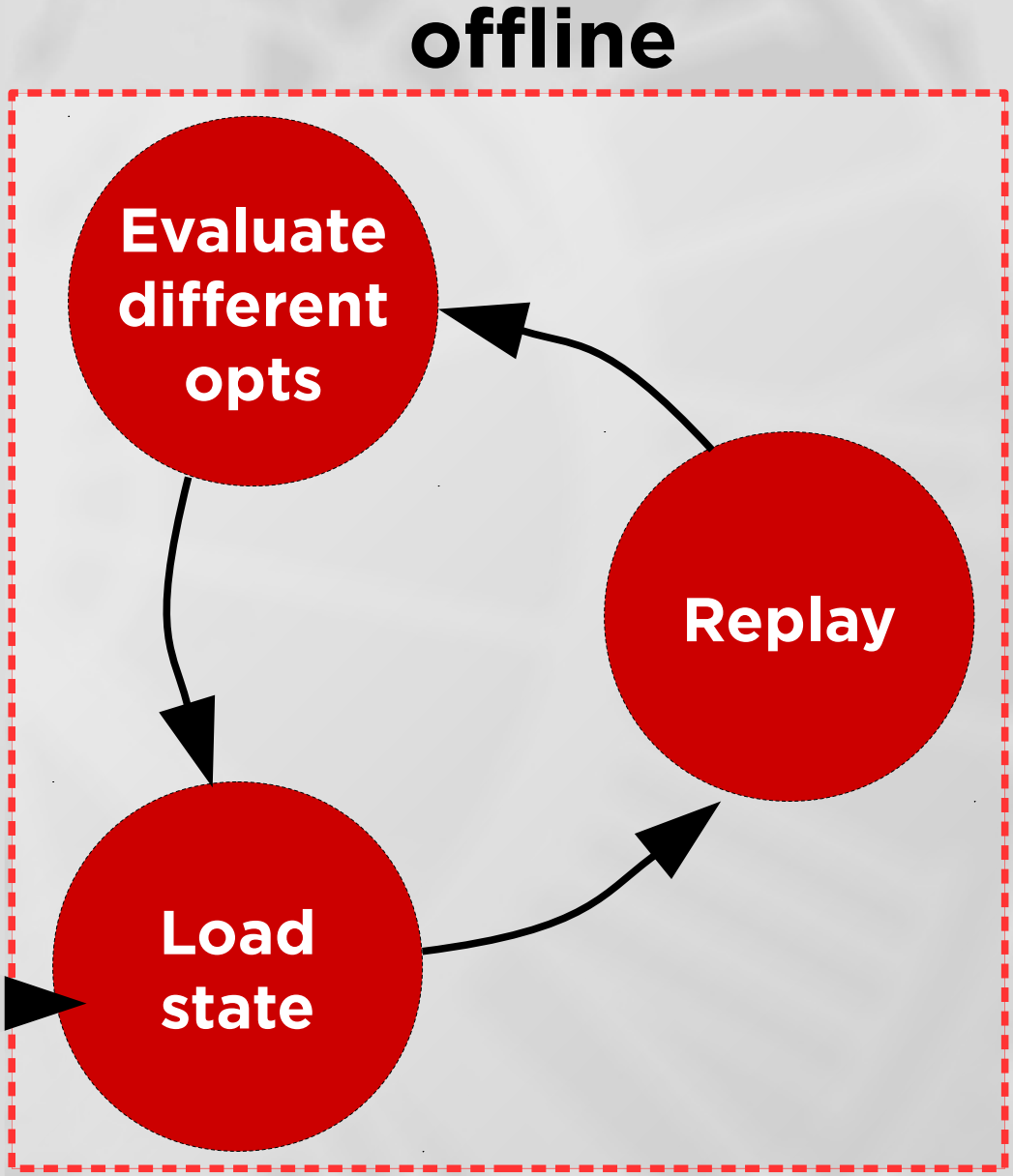
No modification

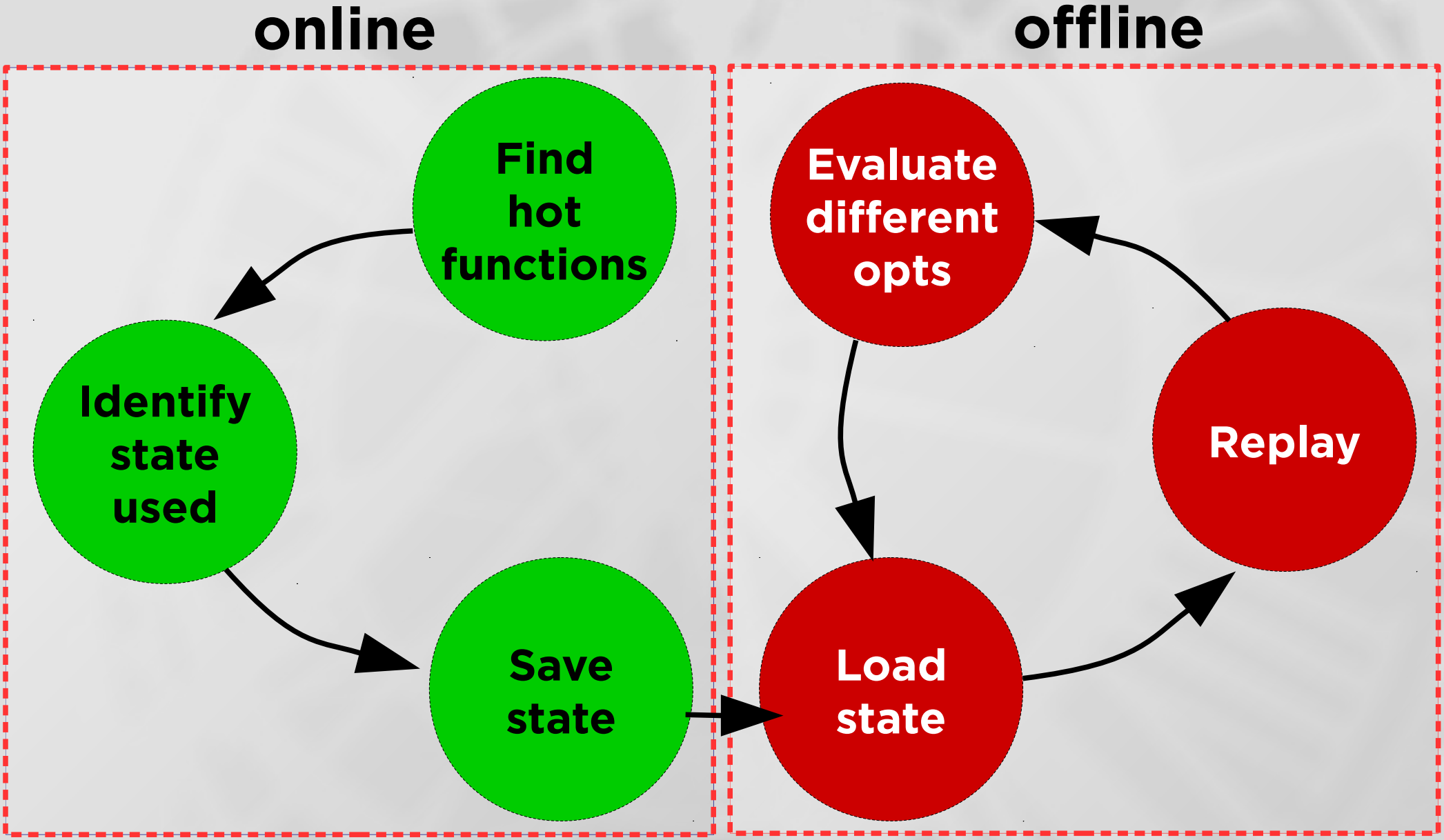
Transparent

Offline replay

No effect on users

Fast evaluation using real input



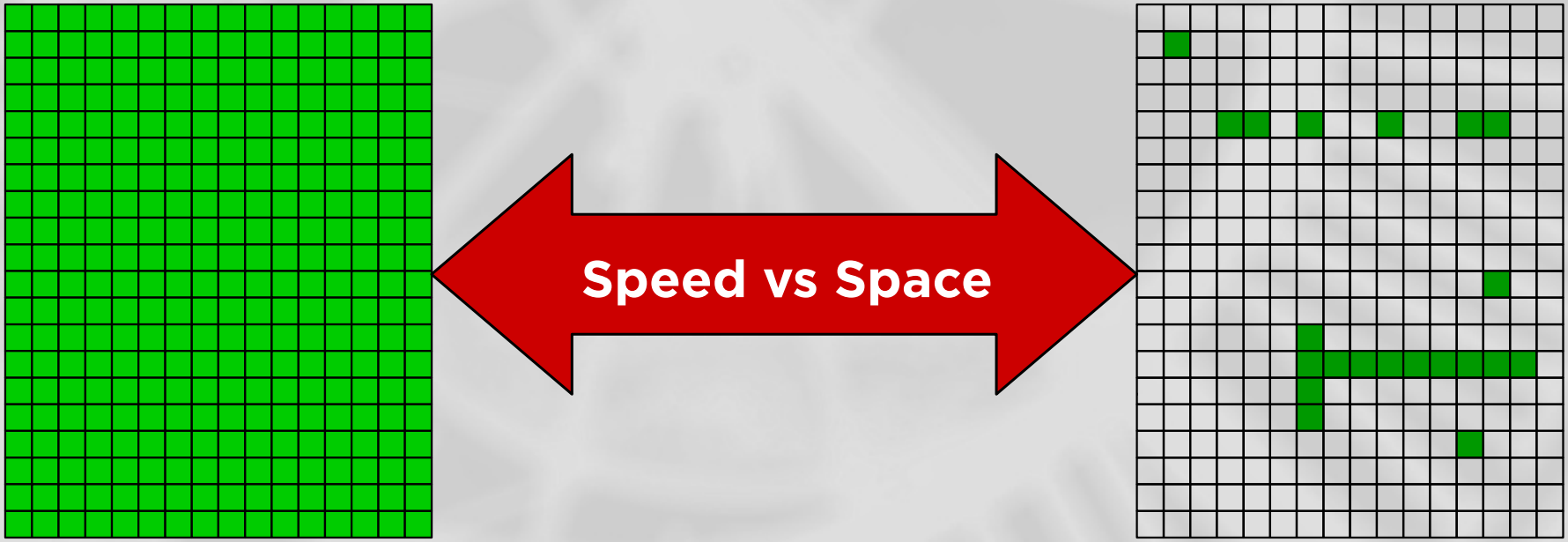

online
offline
Find hot functions
Identify state used
Save state
Evaluate different opts
Replay
Load state

Capture

Existing approaches:

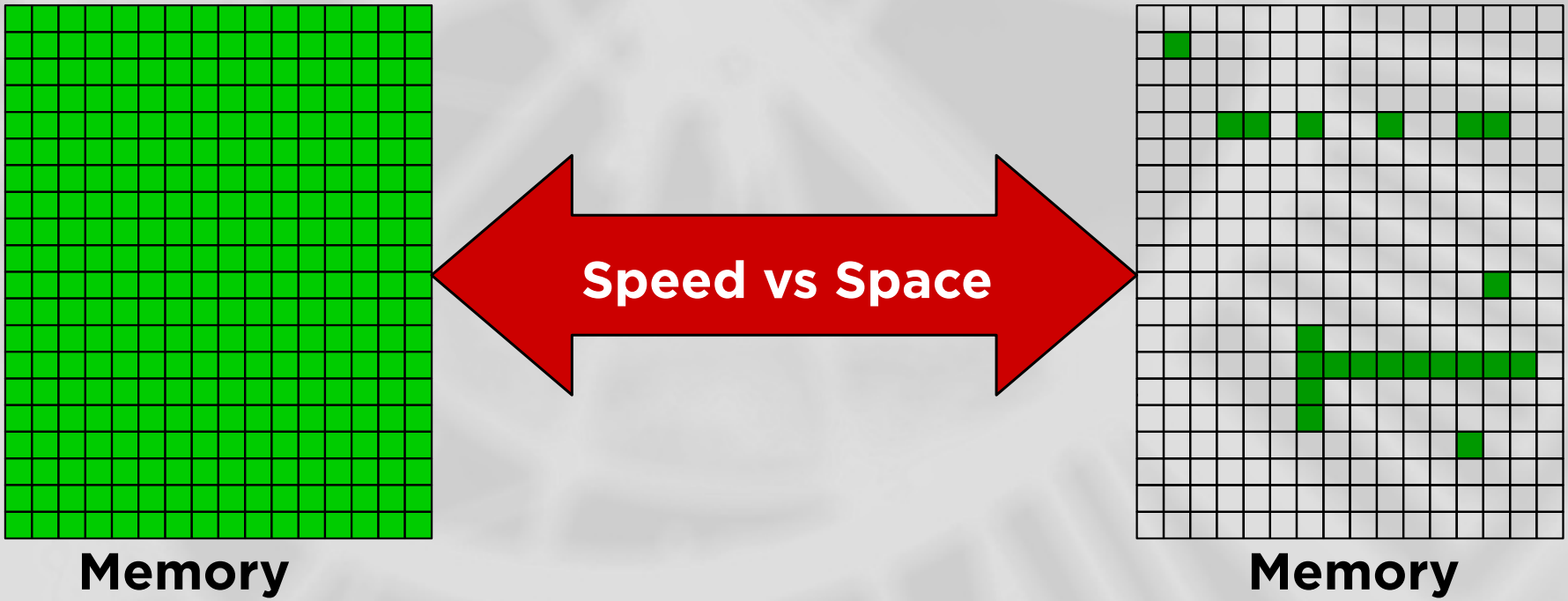
Save everything (quick*)

Save only what's used (slow)



Existing approaches:

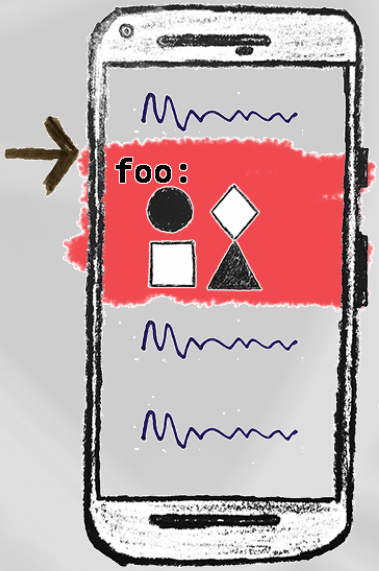
Can't we do it both quick and efficient?



HW already tracks memory accesses

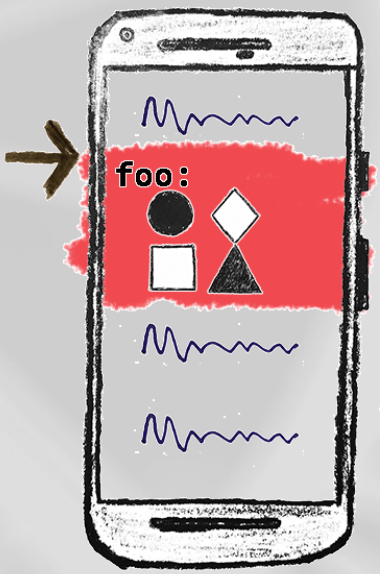
Let's use it

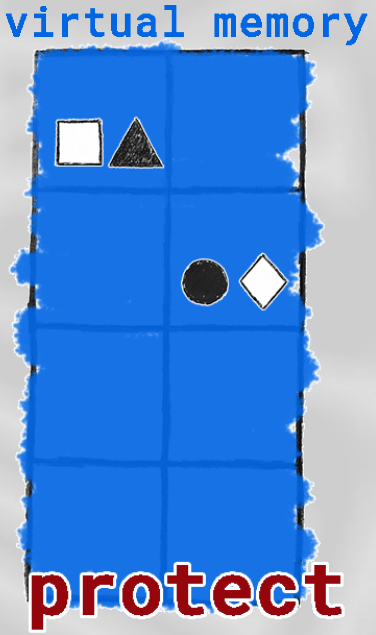
Break at function call



Break at function call

Remove access rights





Break at function call

Remove access rights

SegFault on access

Break at function call

Remove access rights

SegFault on access

User space handler marks used pages

Break at function call

Remove access rights

SegFault on access

User space handler marks used pages

Single SegFault per used page

Store used pages on function exit

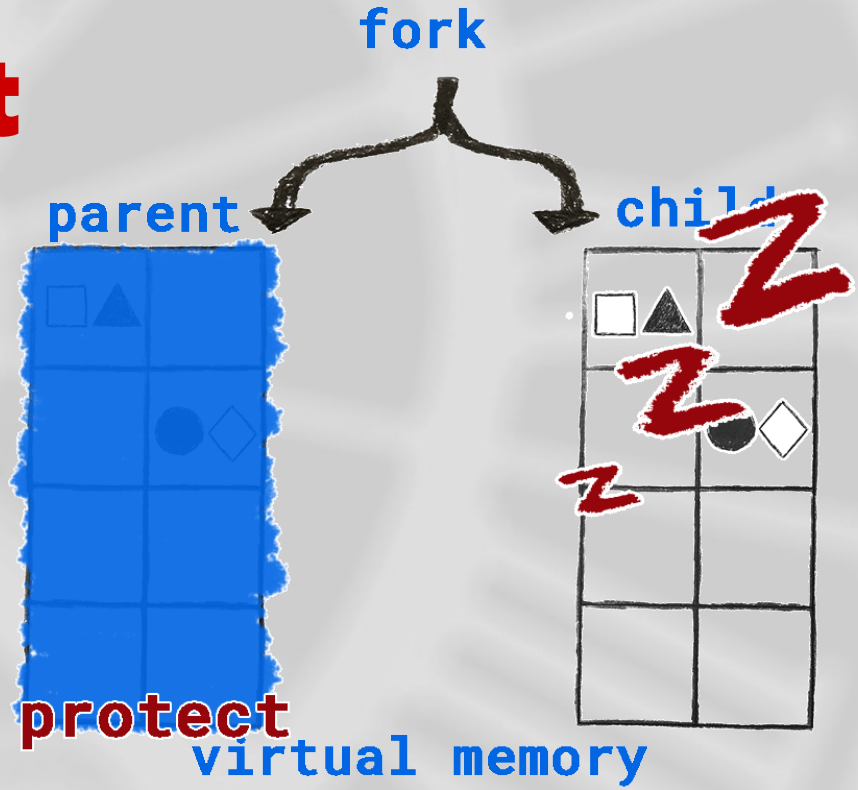
Modified pages?

Should copy everything at function start?

Huge overhead!

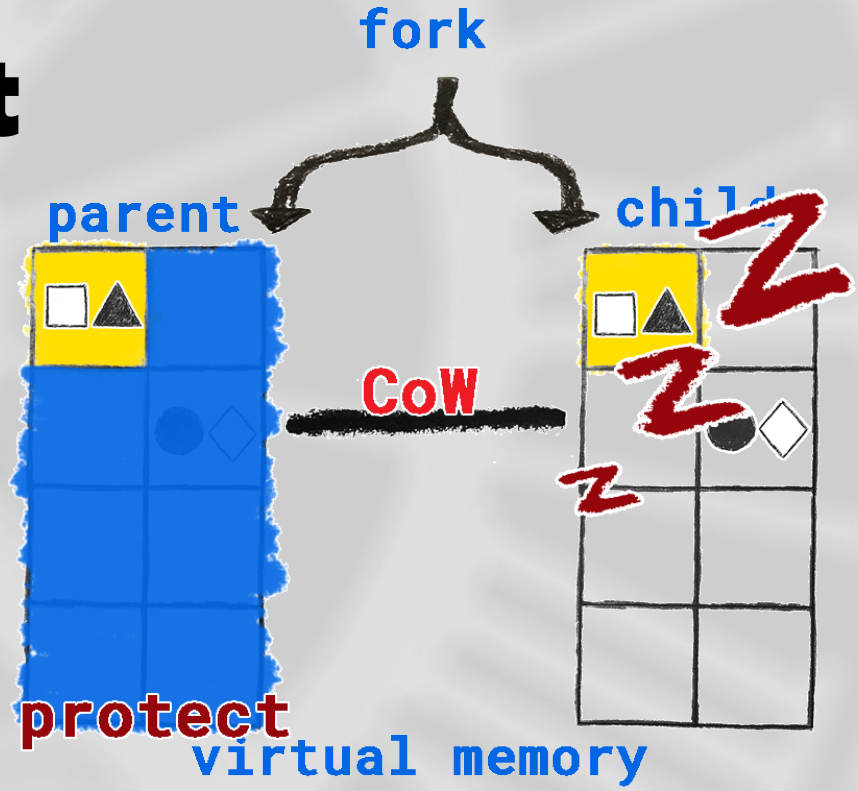
Use fork's CoW!

Fork at function start



Fork at function start

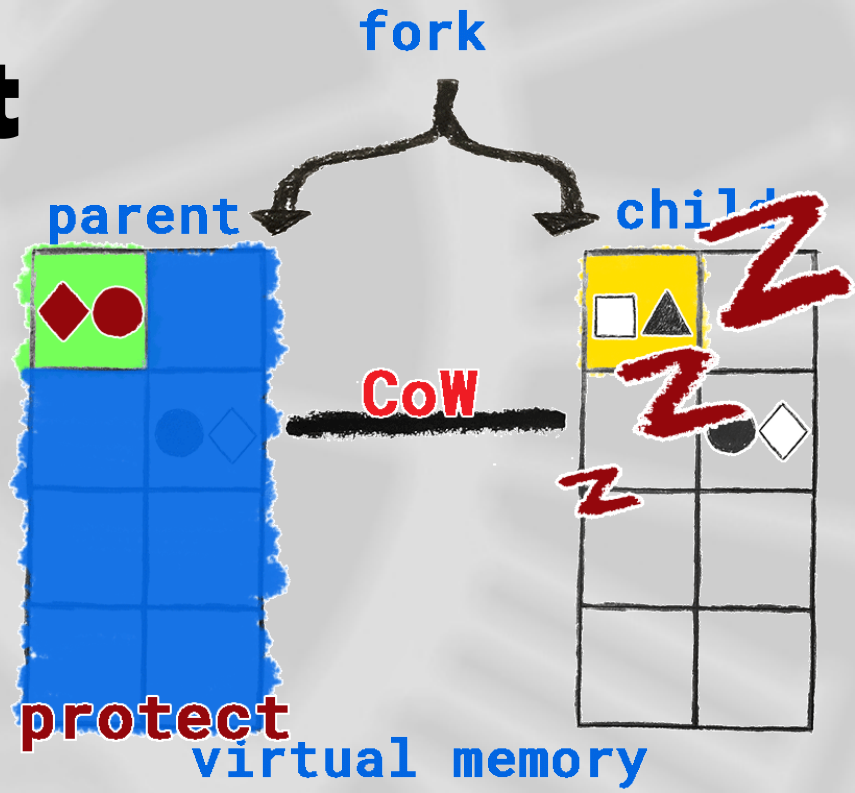
CoW for modified



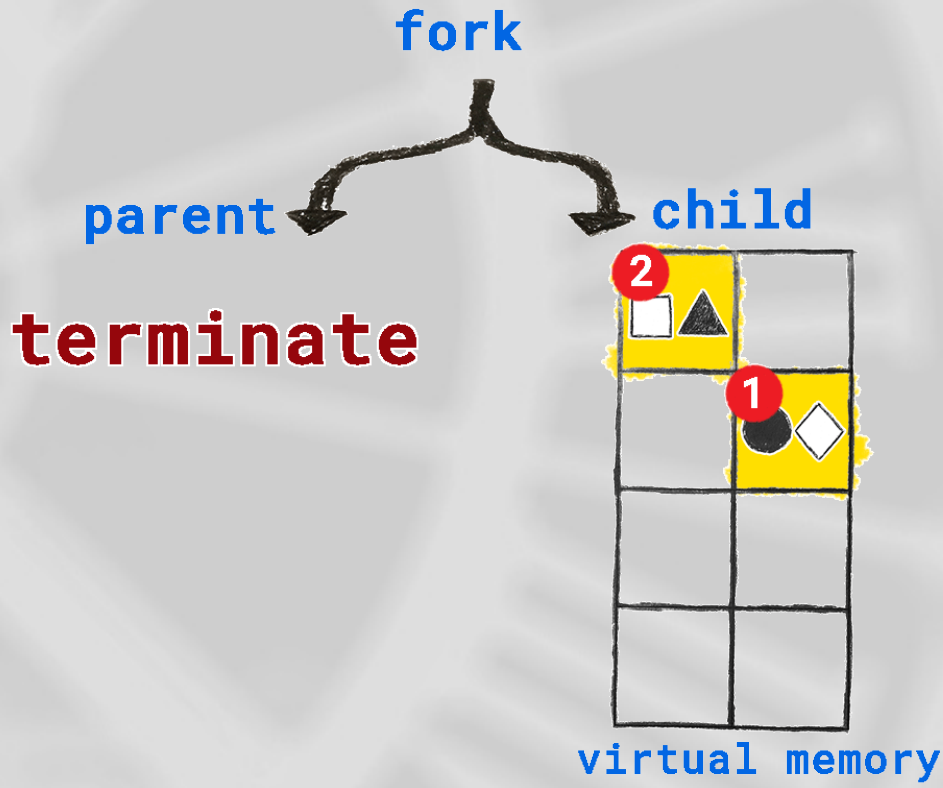
Fork at function start

CoW for modified

Copy at kernel speed



Single CoW per modified page



Low Overhead 

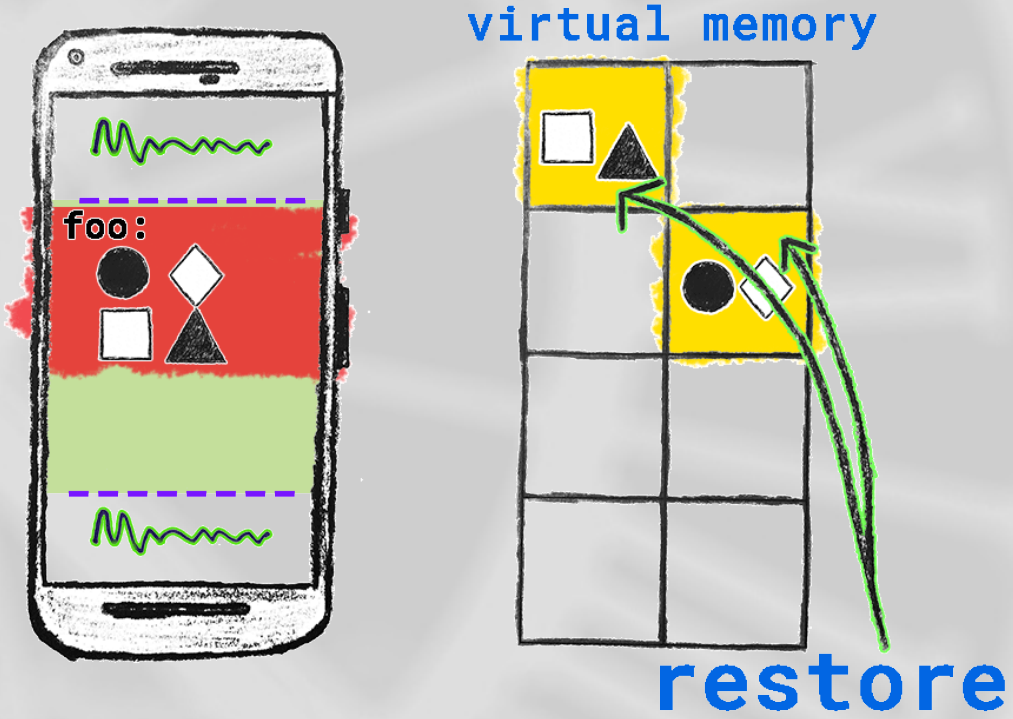
Transparent 

No modifications 

Replay

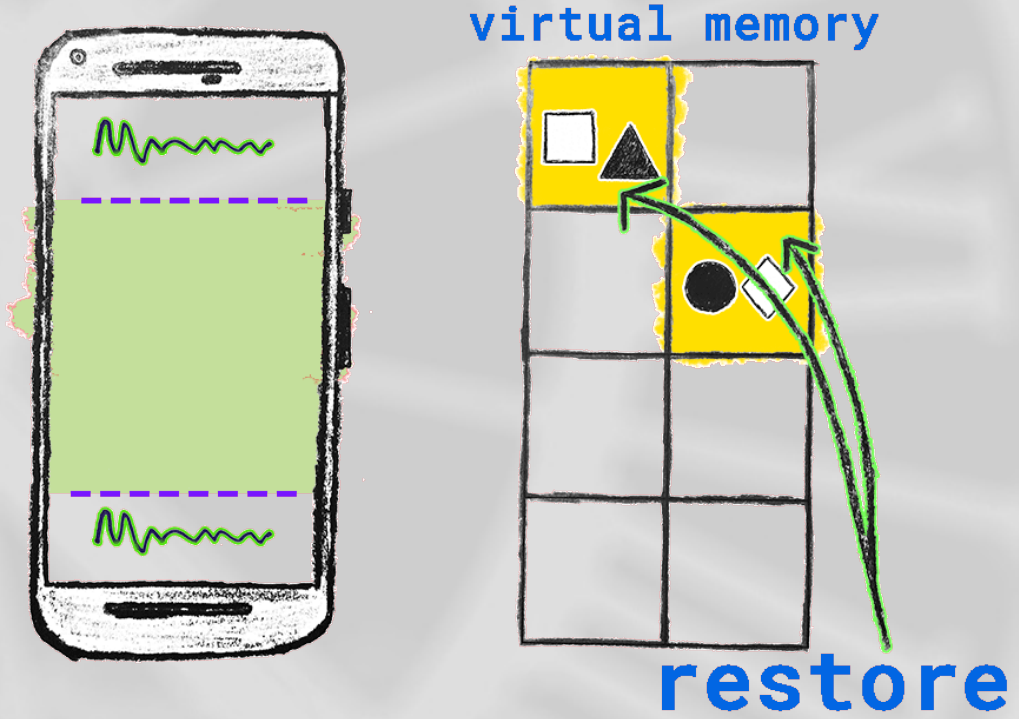
Load state
Load code
Call function

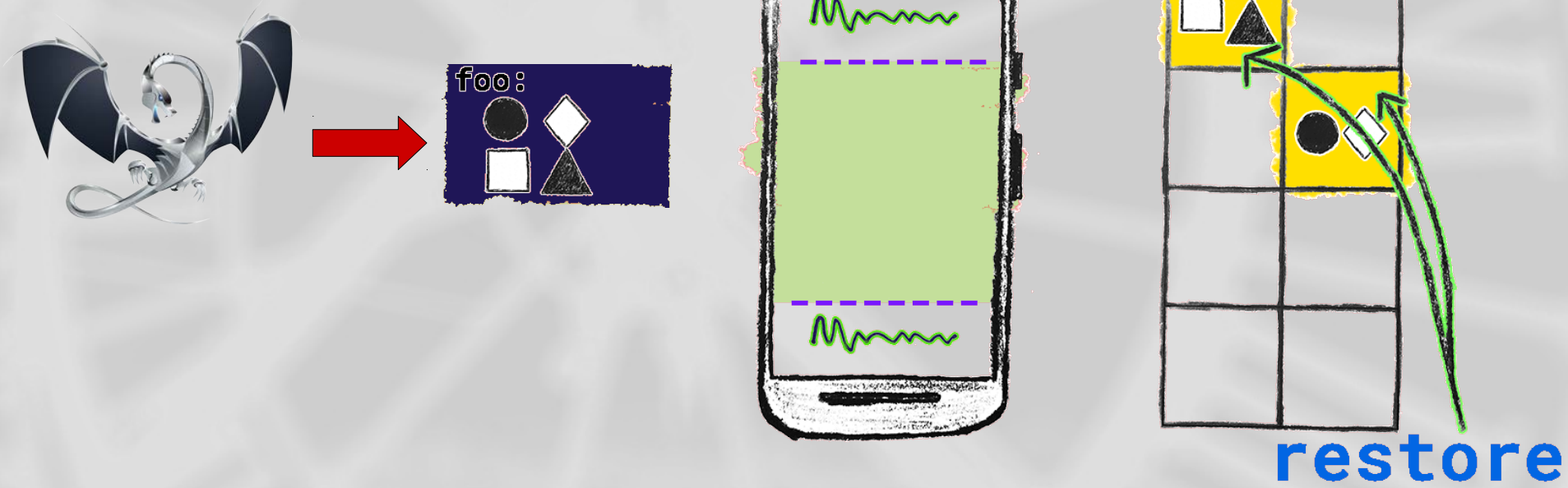
Measure time/energy

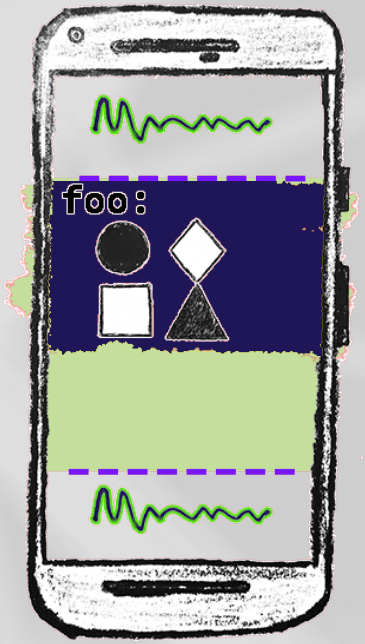


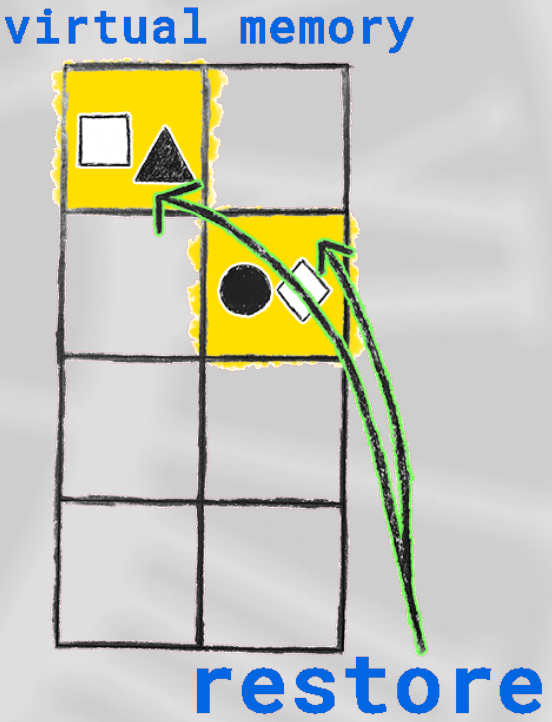
Use Case

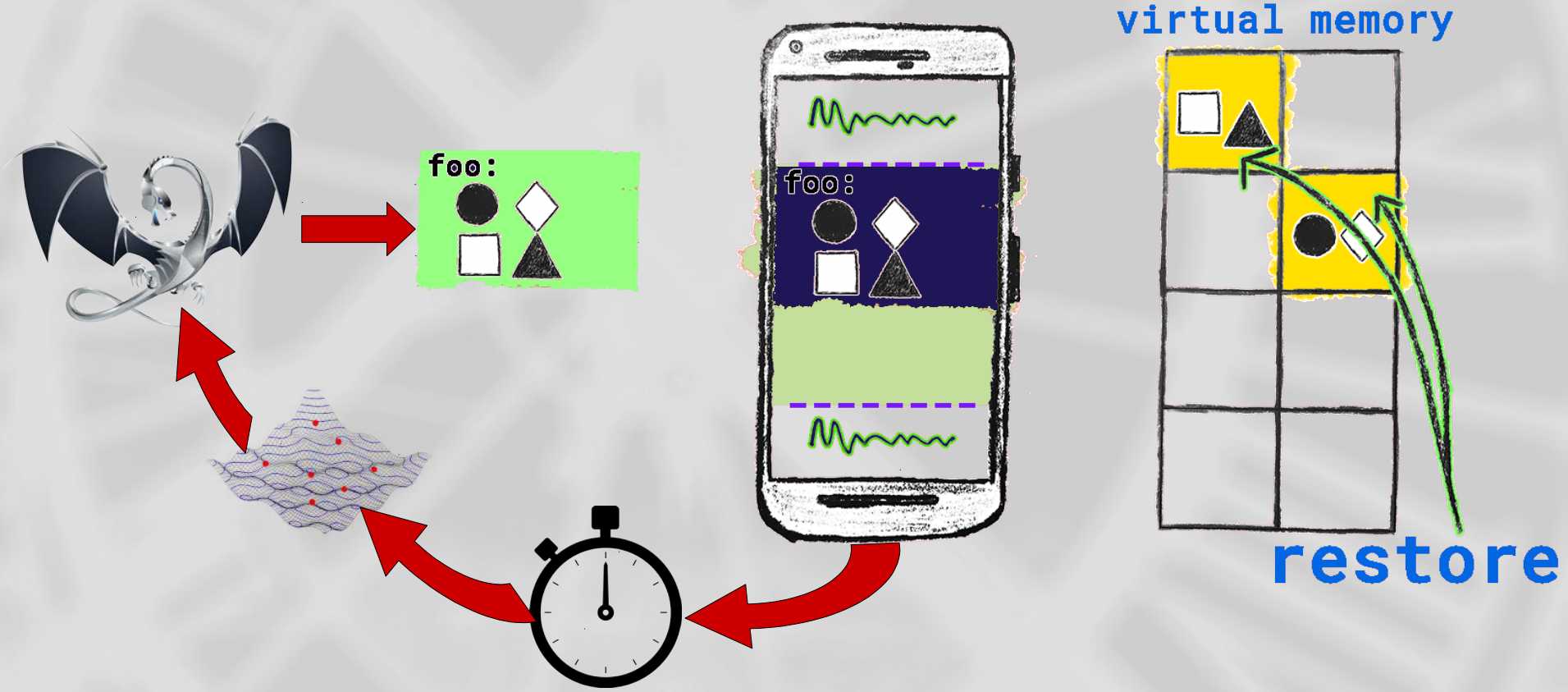
Iterative Compilation


virtual memory
restore

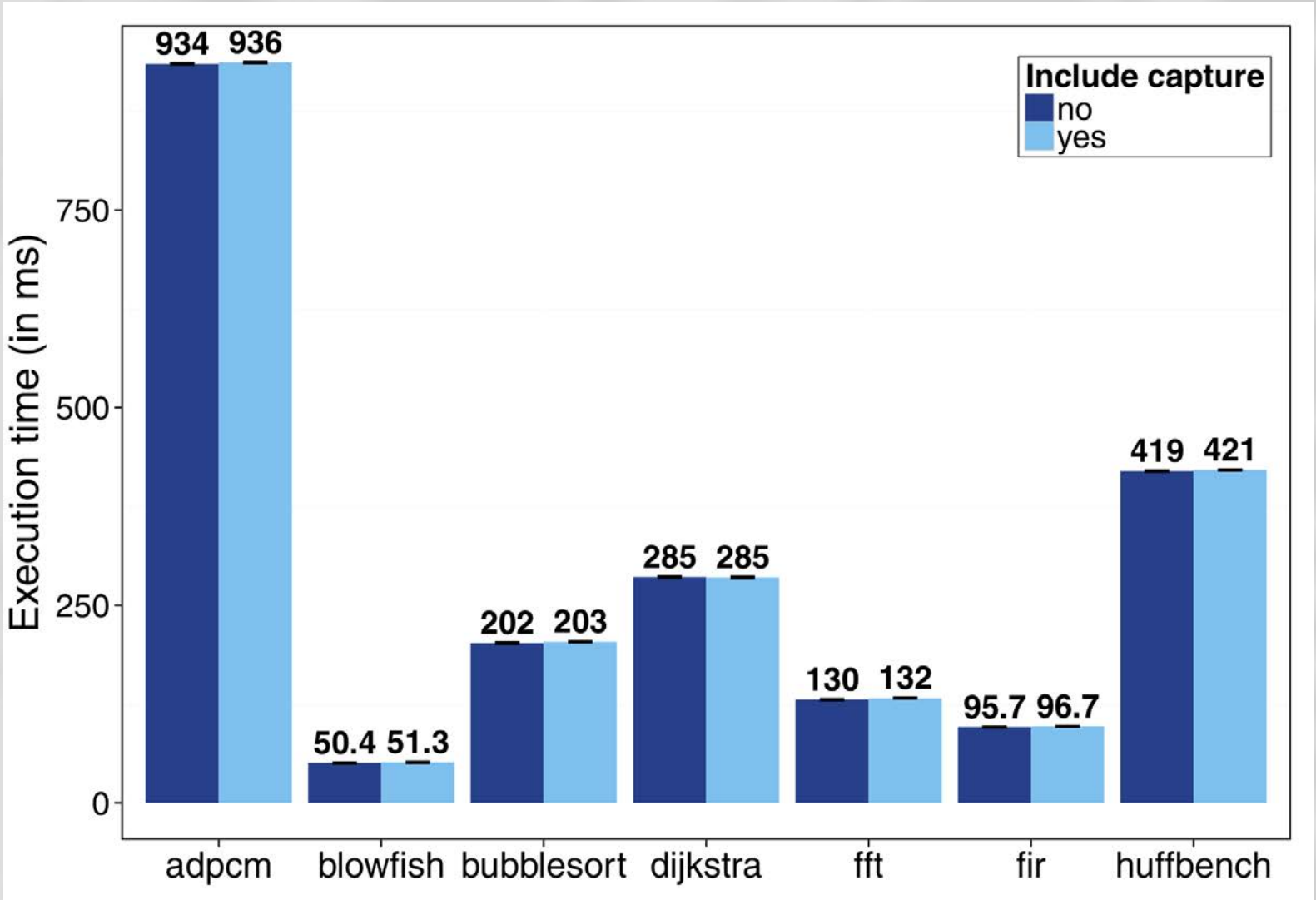

foo:
virtual memory
restore


foo:

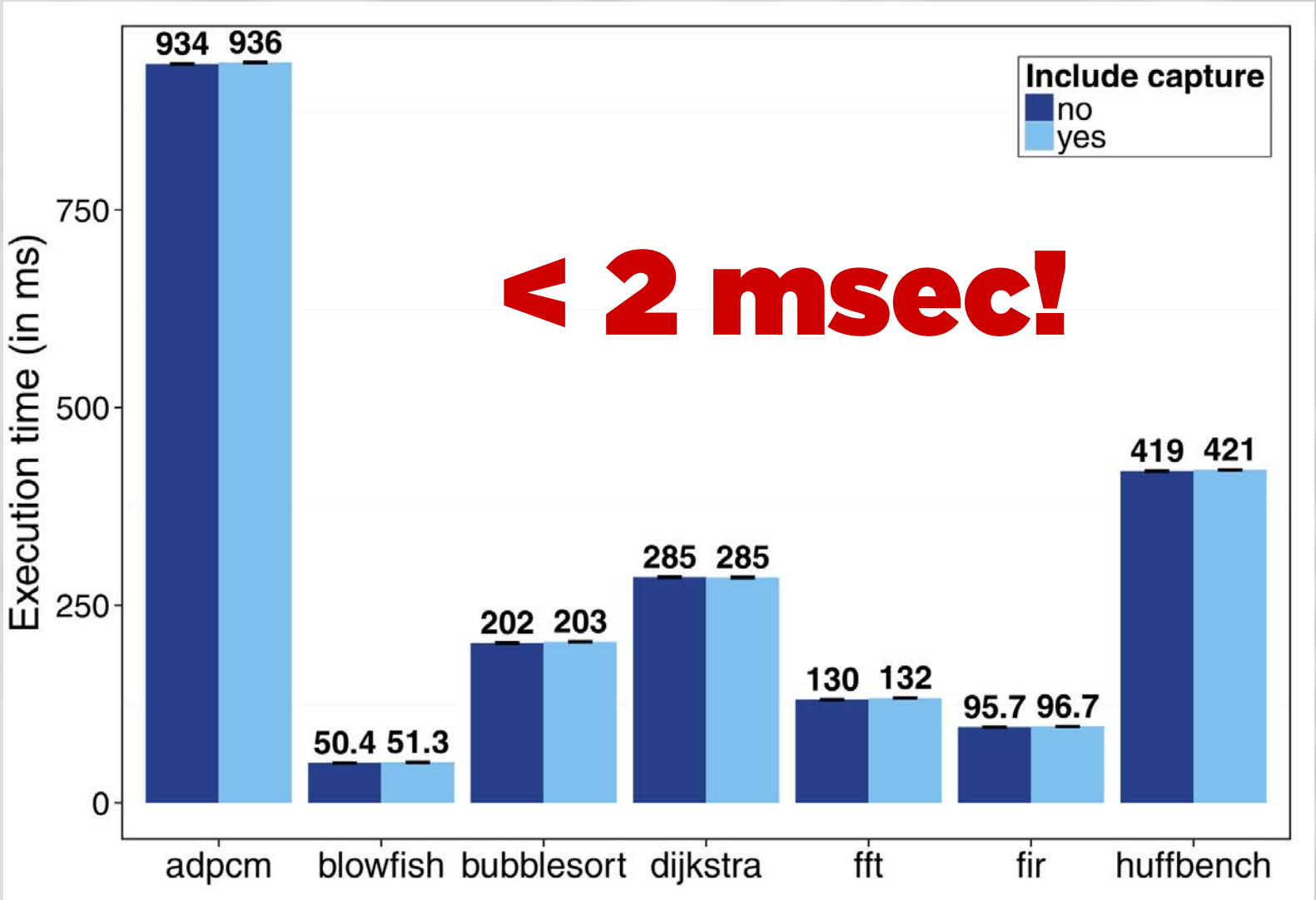

virtual memory
restore


foo:
foo:
virtual memory
restore

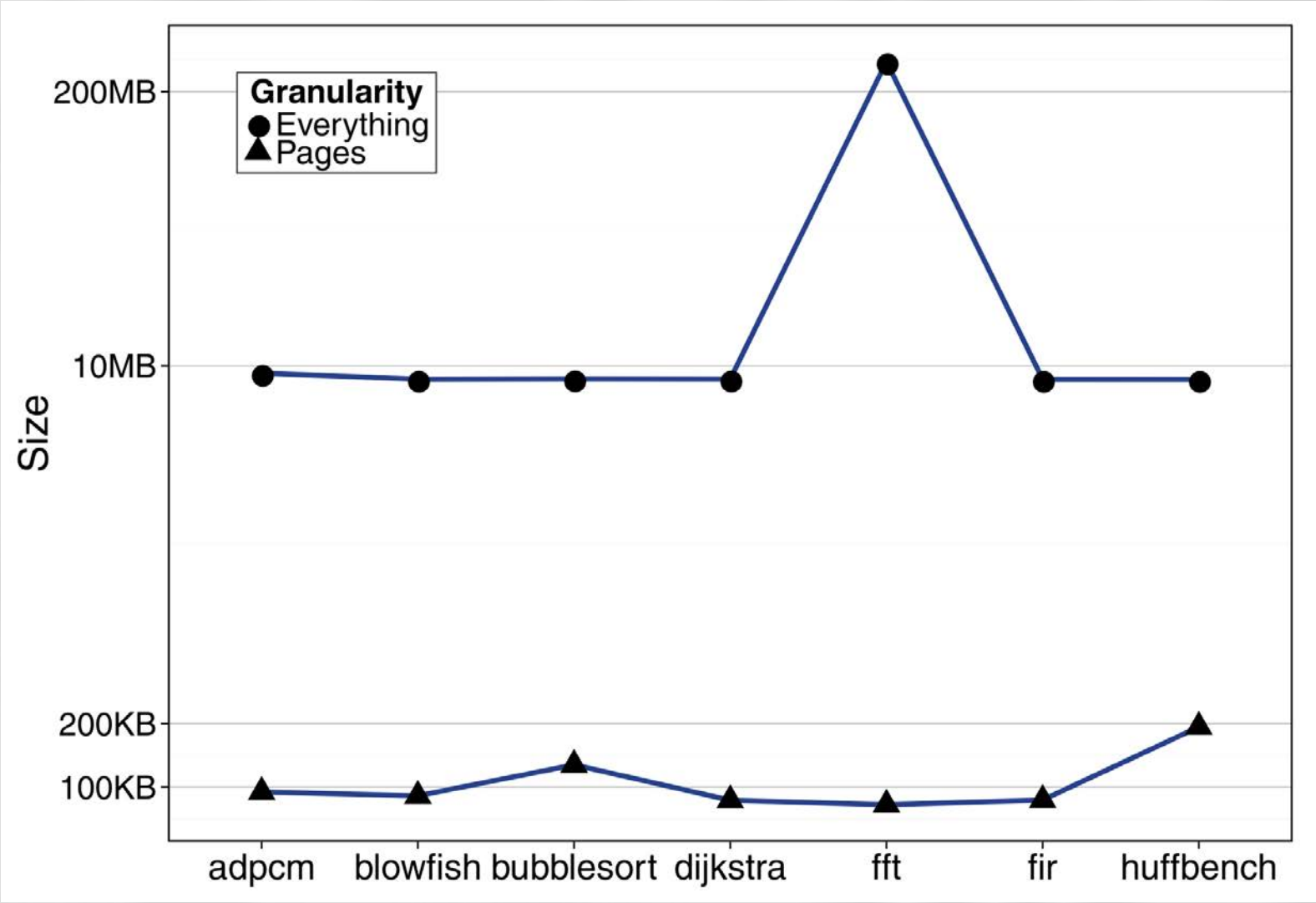
Capture Overhead



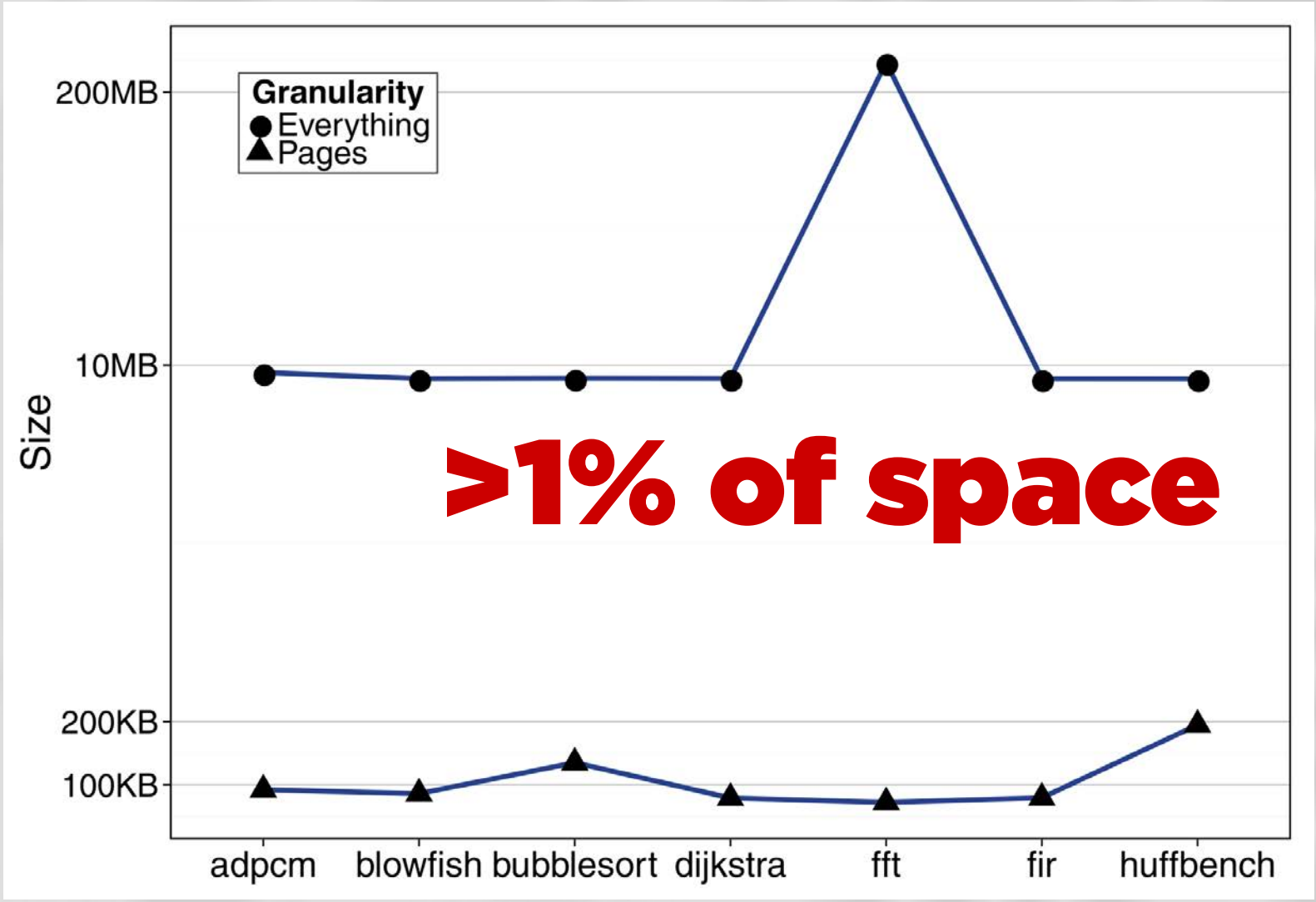
Capture Overhead



Capture Overhead



Capture Overhead



Personalised fast optimisations

Low overhead
Transparent
No modifications

Works for any app

Real inputs

Reliable Optimisations

Summary

We need the 3 Rs

Real
Representative
Reproducible

1st technique:

User-centric metrics

2nd technique:

Personalised optimisation

Easy to generate workloads
Easy to use them

Real optimisations for real people

Backup slides

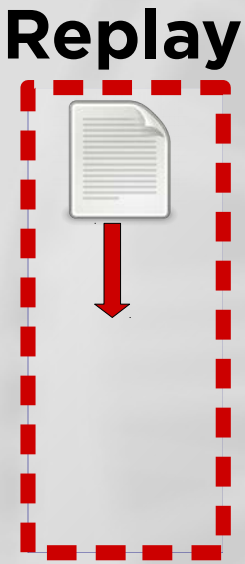
After the fact

Lag End Estimation

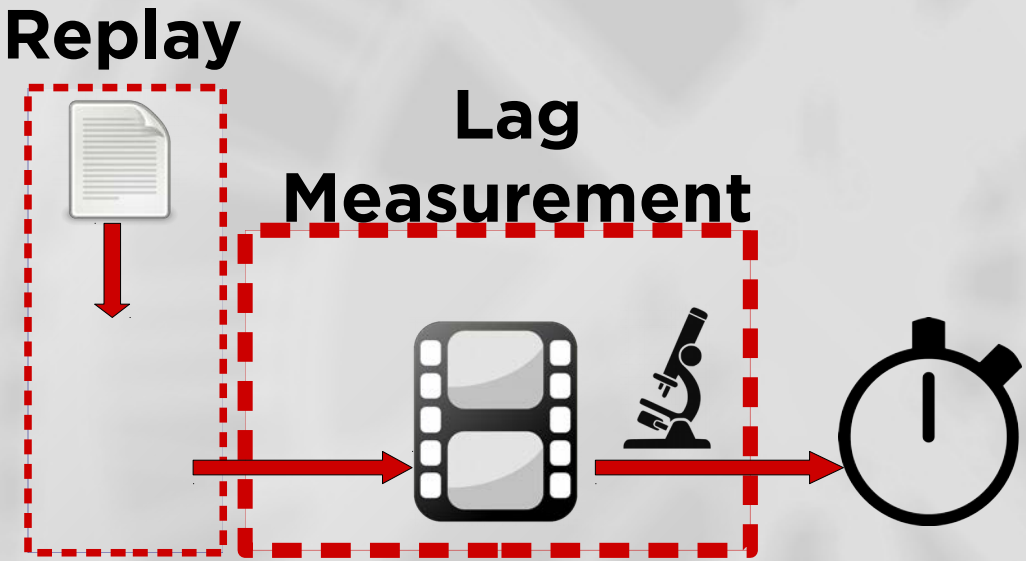
Can only identify lags offline

Want to identify lags online

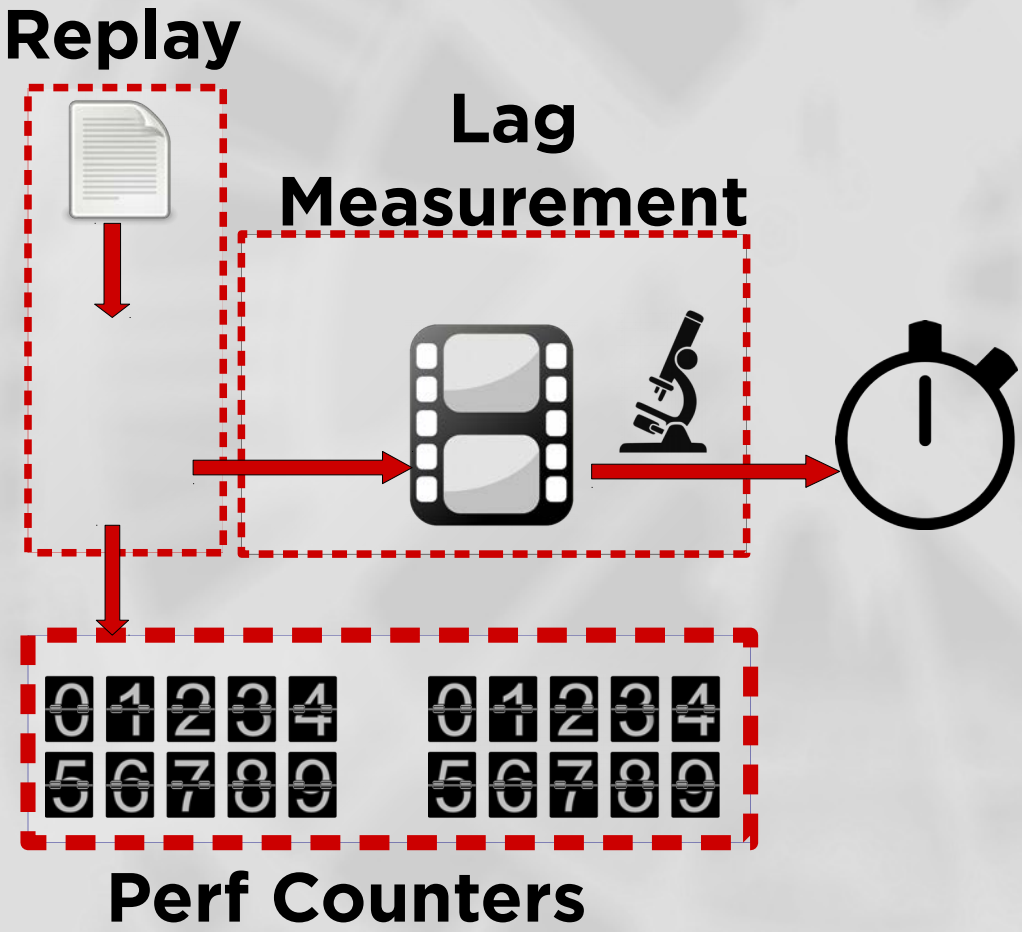
Training



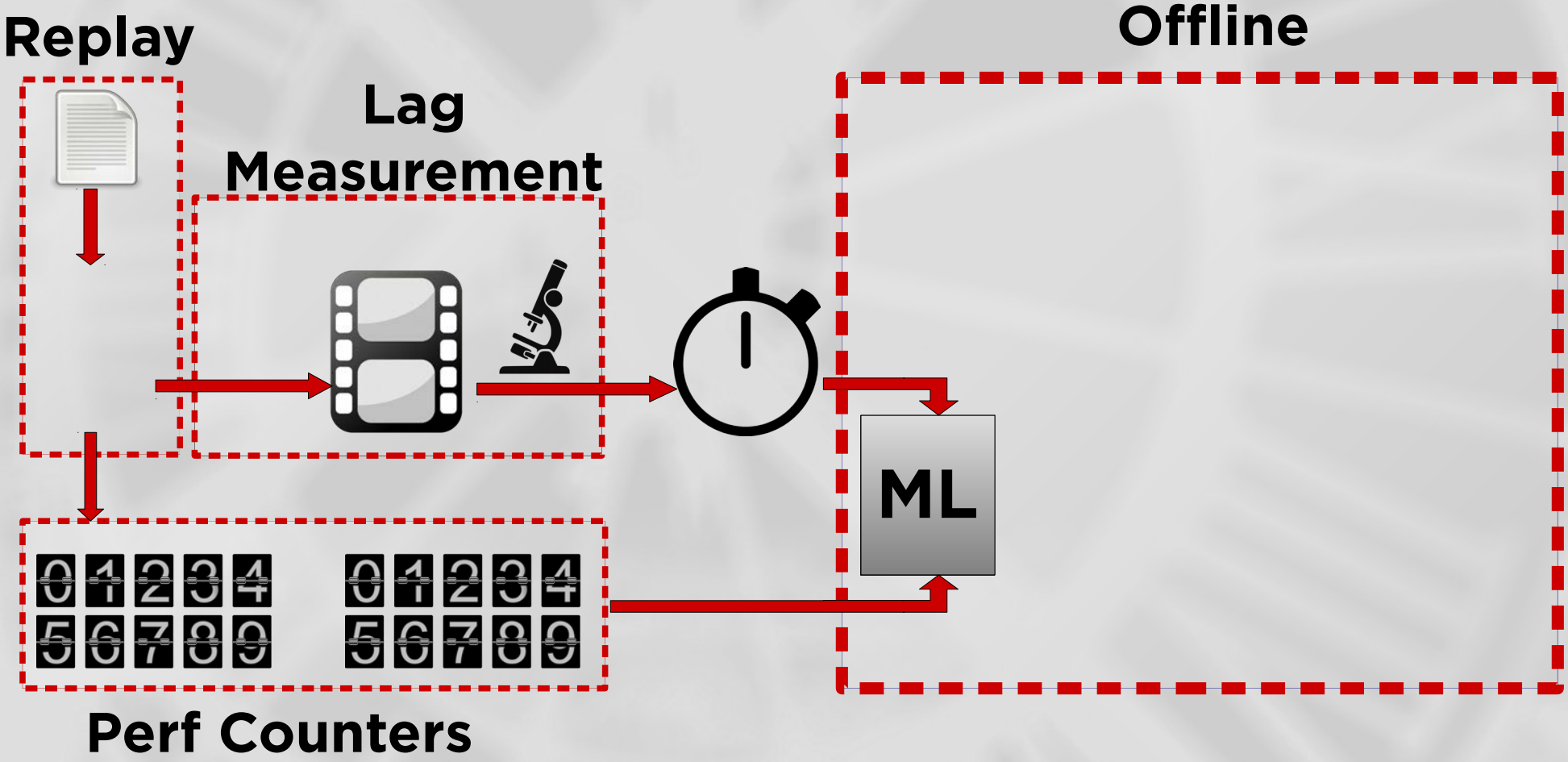
Training



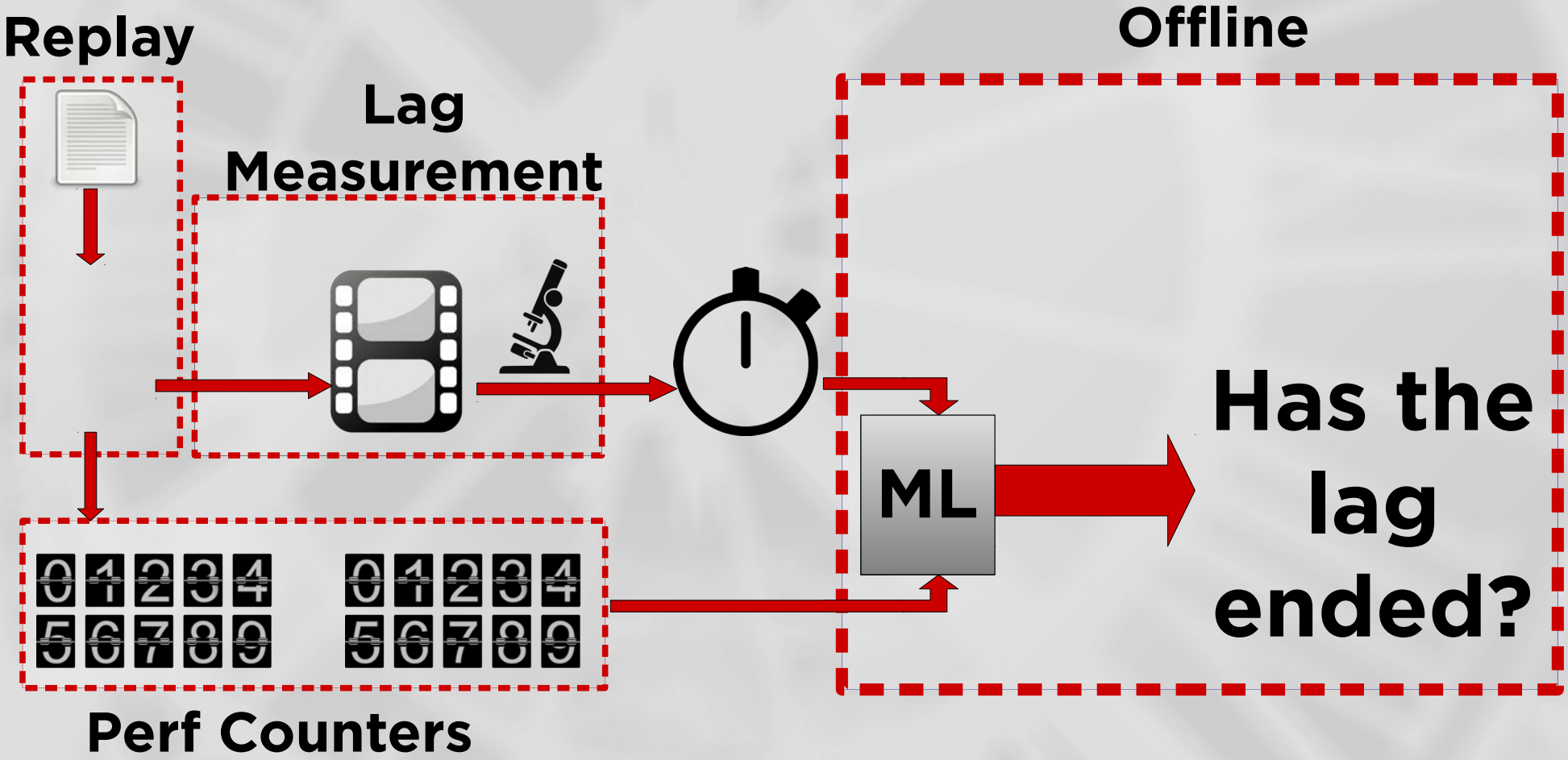
Training



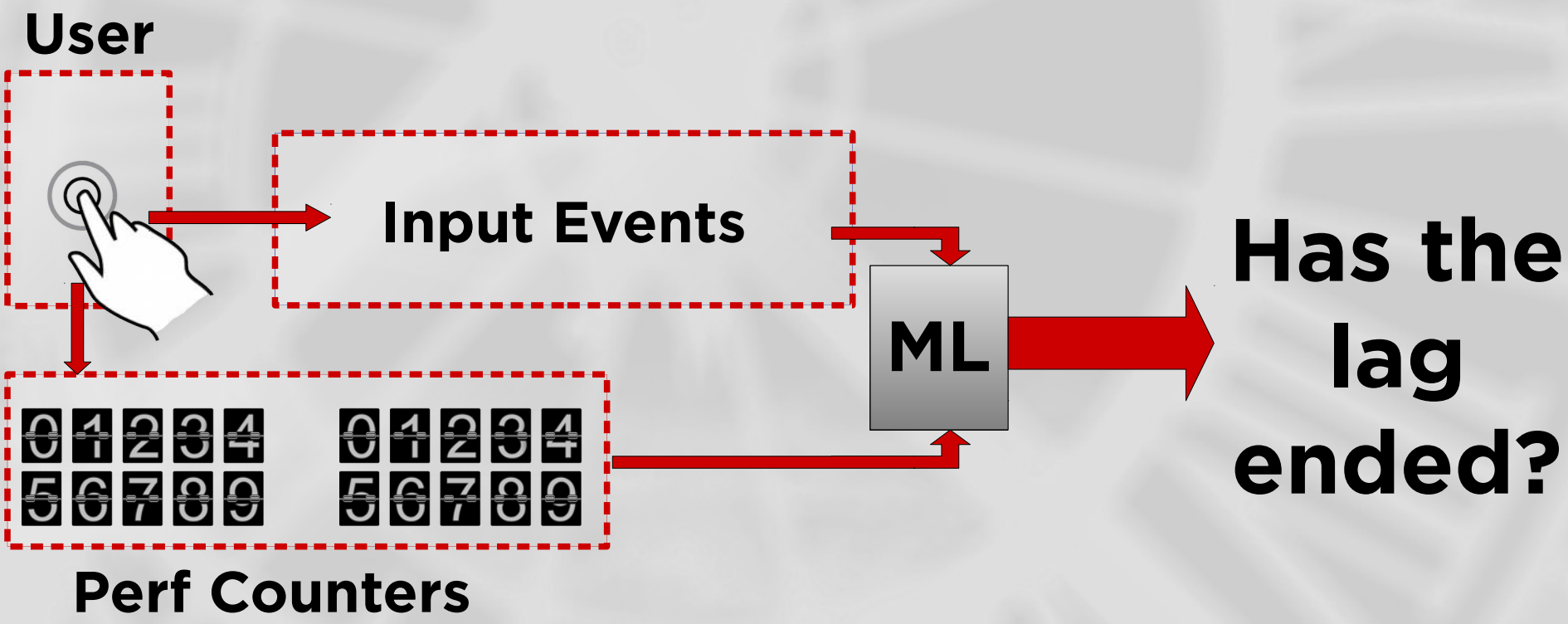
Training



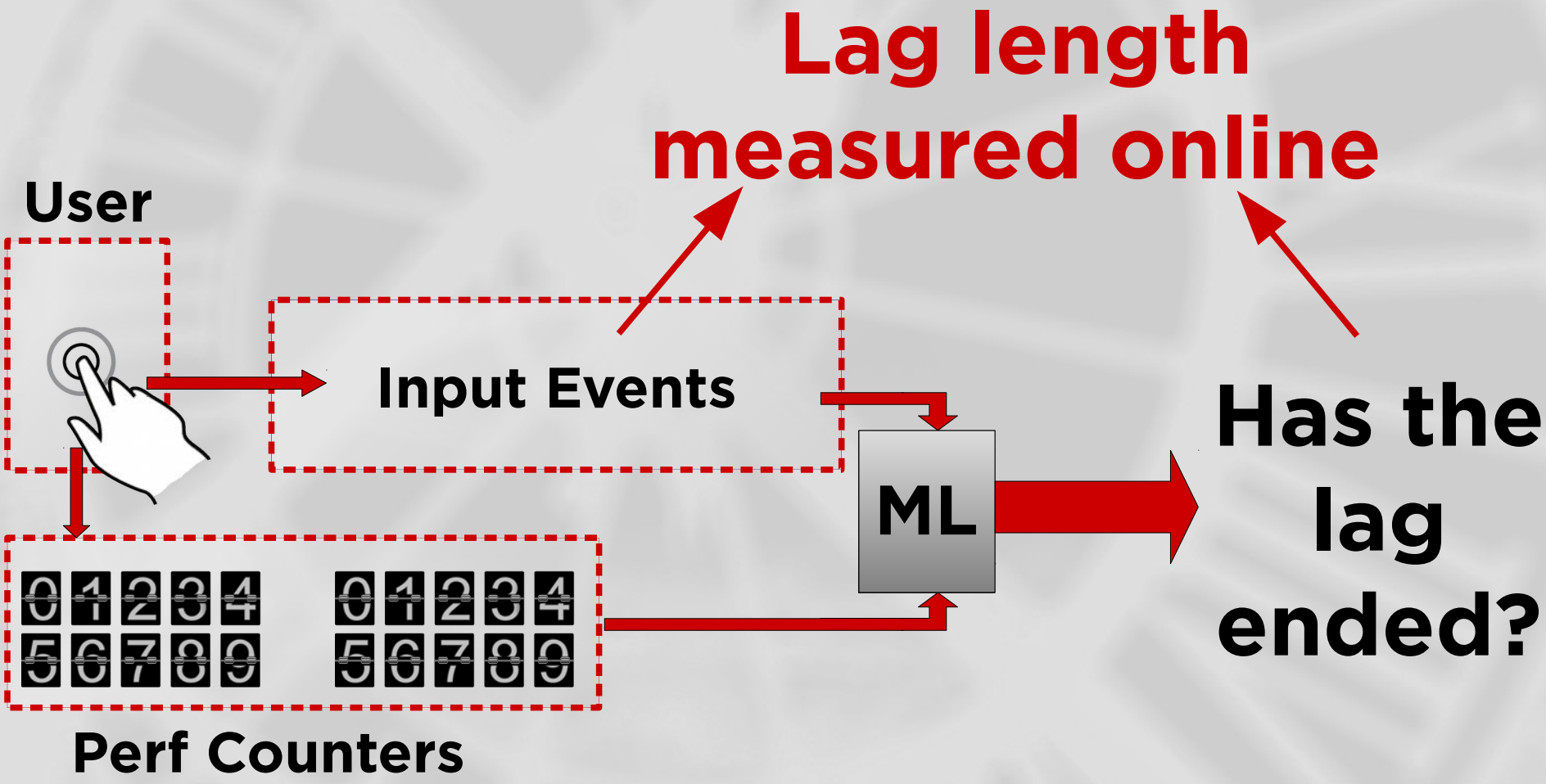
Training


Replay
Lag Measurement
Offline
ML
Has the lag ended?
Perf Counters

Online estimation



Online prediction



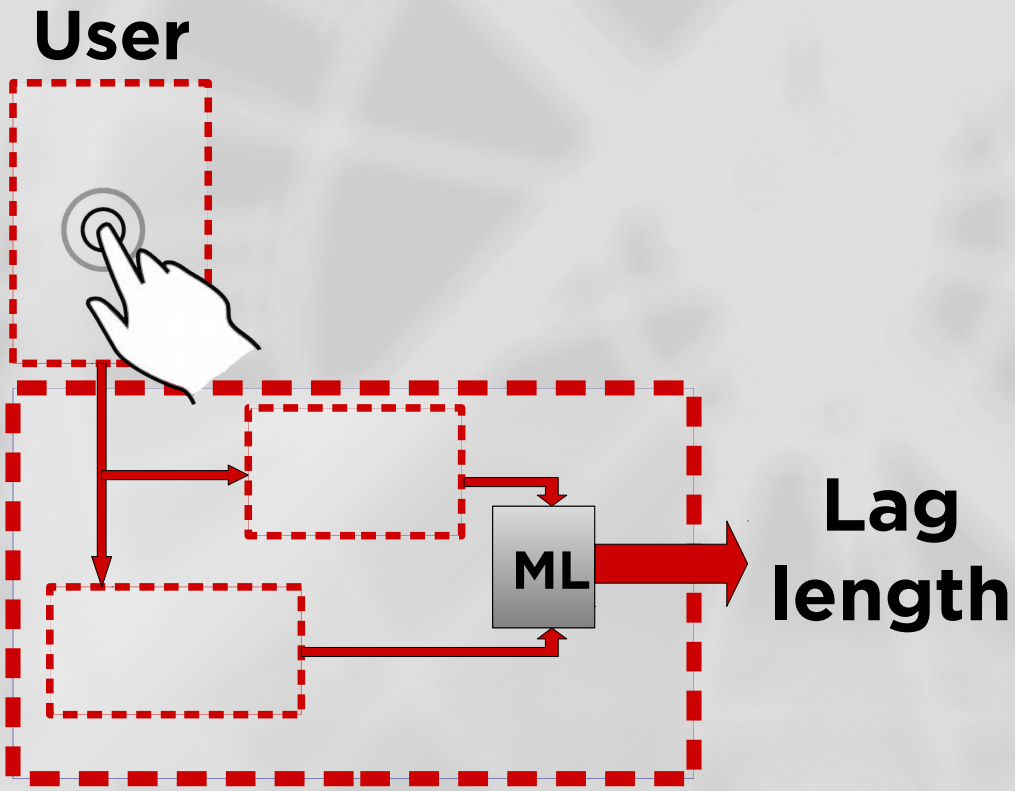
Before the fact

Lag End Predictor

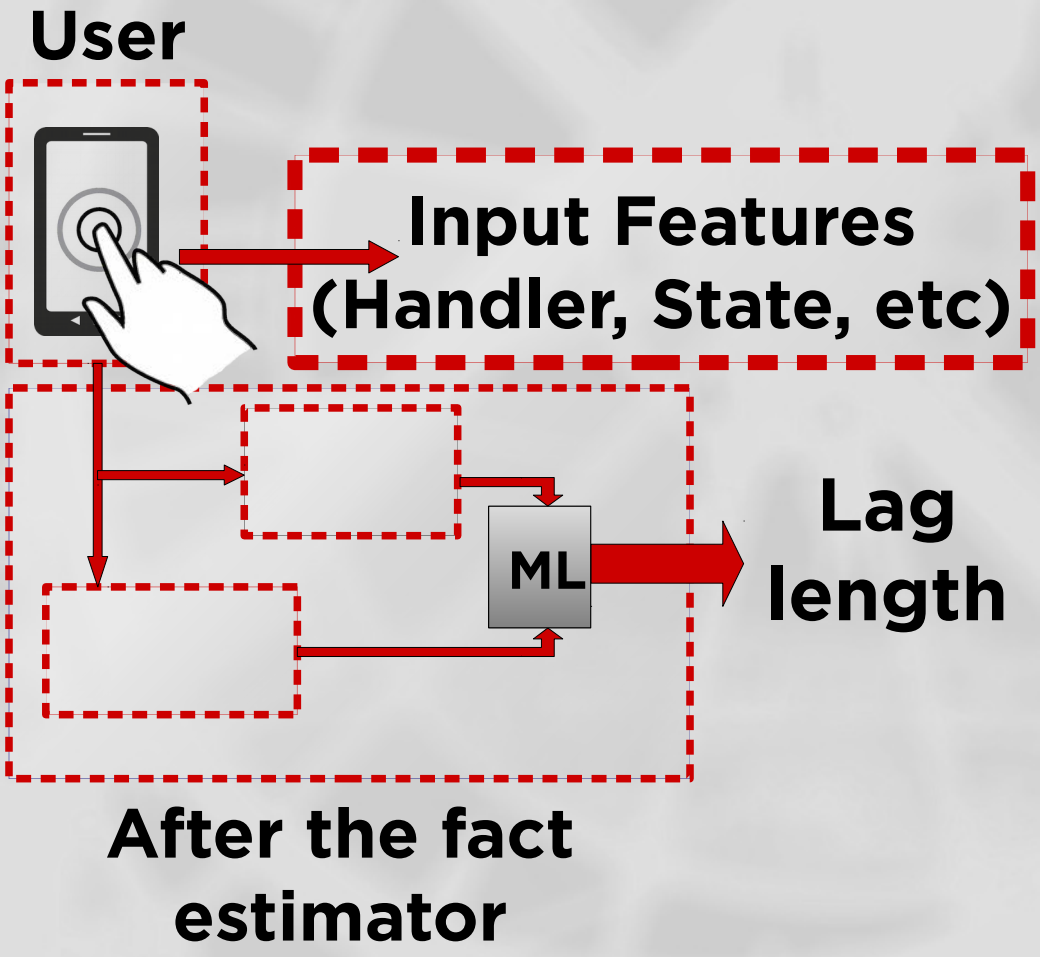
Can tell whether the interaction has ended

Want to tell when the interaction will end

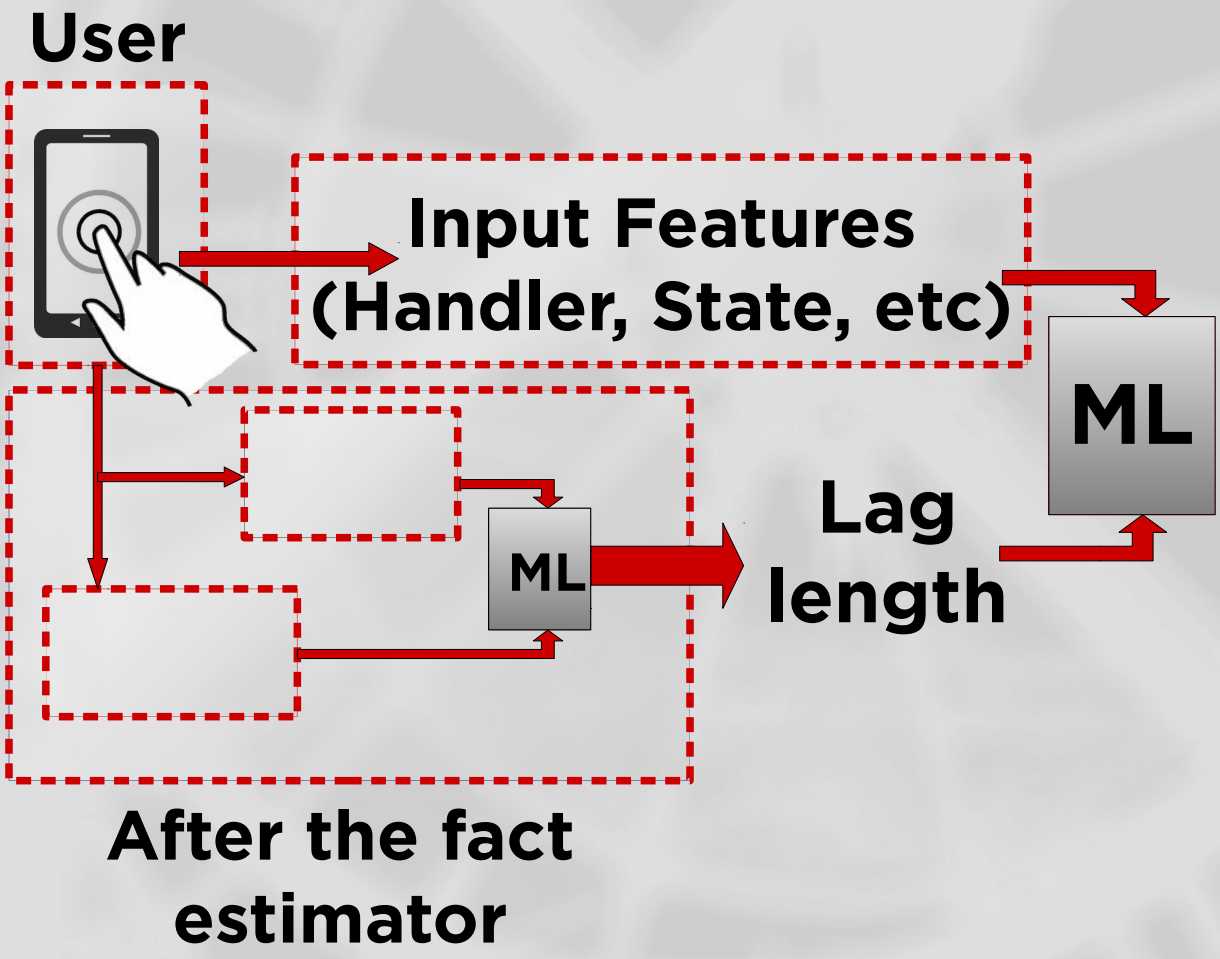
Lag end prediction



Lag end prediction



Lag end prediction



Lag end prediction