# Measuring Energy Consumption of Software


Akademy 2023

Volker Krause

vkrause@kde.org
@VolkerKrause@floss.social

# Why do we care?

- Battery powered devices
  - Less energy use = longer device uses
- Performance
  - Less energy use = less thermal throttling
- Climate impact
  - See one of Joseph’s talk on just how big that is for IT

# Software impact on energy consumption

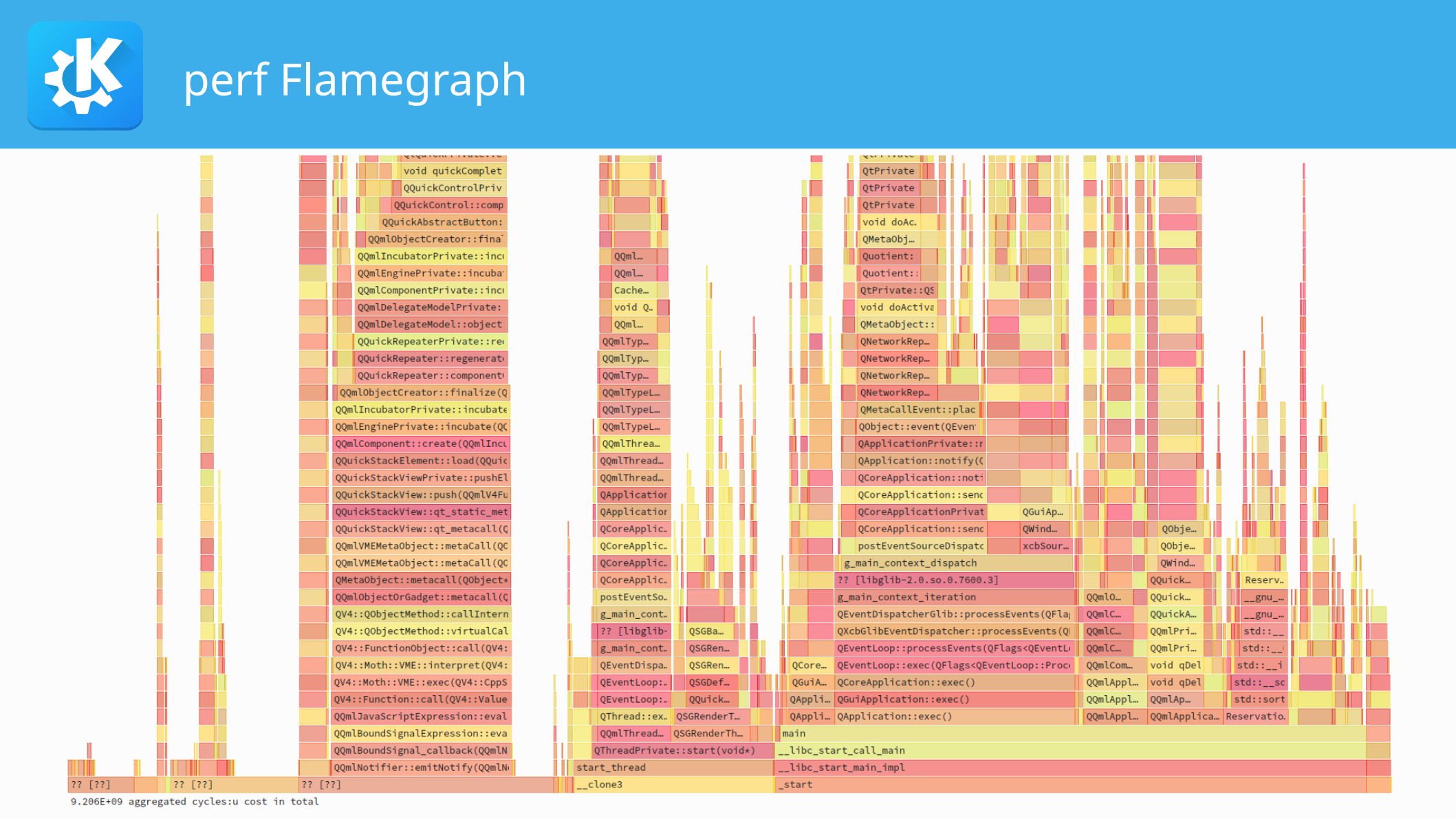
- Use of hardware resources during workloads
  - CPU, GPU, memory, I/O, etc
- Hardware power management settings
  - CPU frequency, screen brightness, switching off unused things
- Idle behavior
  - Wakeups, continuously running timers

# Does this make a difference?

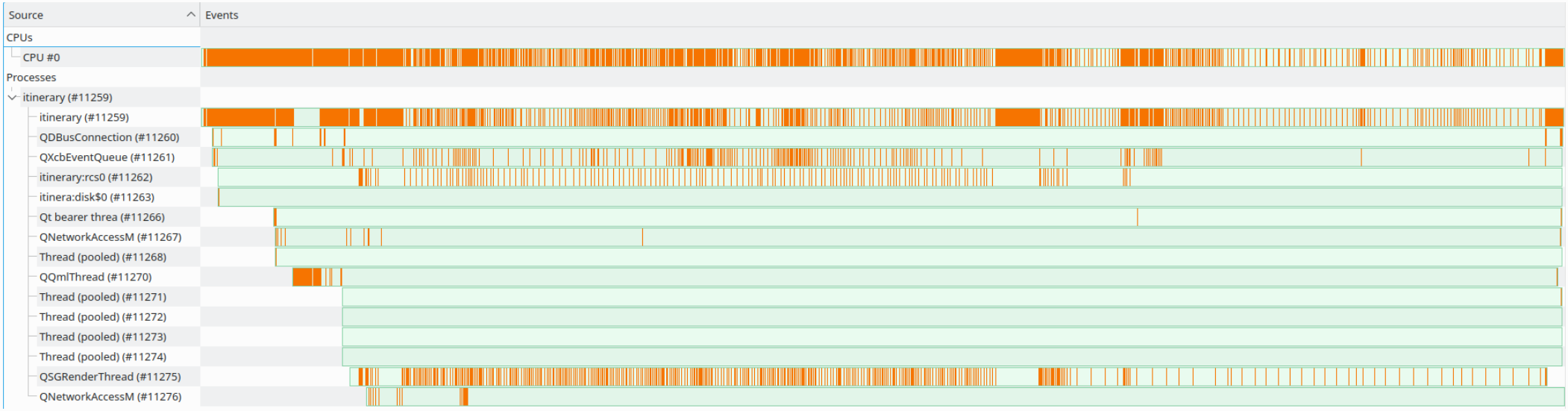
- Physics isn’t linear
  - Gains can be higher than intuitively assumed
- Scaling effects
  - Optimizations multiply over time and by number of users

# Understanding what our software does

- General purpose profilers
  - `perf record --call-graph=dwarf <app>`
  - Hotspot flamegraph for workloads
  - Hotspot timeline for idle behavior
- Specialized tools
  - powertop for idle behavior


perf Flamegraph
void quickComplet
QQuickControlPriv
QQuickControl::comp
QQuickAbstractButton:
QQmlObjectCreator::fina
QQmlIncubatorPrivate::inc
QQmlEnginePrivate::incuba
QQmlComponentPrivate::inc
QQmlDelegateModelPrivate:
QQmlDelegateModel::object
QQuickRepeaterPrivate::re
QQuickRepeater::regenerat
QQuickRepeater::component
QQmlObjectCreator::finalize(Q
QQmlIncubatorPrivate::incubate
QQmlEnginePrivate::incubate(QC
QQmlComponent::create(QQmlIncu
QQuickStackElement::load(QQuic
QQuickStackViewPrivate::pushEl
QQuickStackView::push(QQmlV4Fu
QQuickStackView::qt_static_met
QQuickStackView::qt_metacall(Q
QQmlVMEMetaObject::metaCall(QC
QQmlVMEMetaObject::metaCall(QC
QMetaObject::metacall(QObject*
QQmlObjectOrGadget::metacall(Q
QV4::QObjectMethod::callIntern
QV4::QObjectMethod::virtualCal
QV4::FunctionObject::call(QV4:
QV4::Moth::VME::interpret(QV4:
QV4::Moth::VME::exec(QV4::CppS
QV4::Function::call(QV4::Value
QQmlJavaScriptExpression::eval
QQmlBoundSignalExpression::eva
QQmlBoundSignal_callback(QQmlN
QQmlNotifier::emitNotify(QQmlN
?? [??]
?? [??]
?? [??]
QQml…
QQml…
Cache…
void Q…
QQml…
QQmlTyp…
QQmlTyp…
QQmlTyp…
QQmlTypeL…
QQmlTypeL…
QQmlTypeL…
QQmlThrea…
QQmlThread…
QQmlThread…
QApplicatior
QApplicatior
QCoreApplic…
QCoreApplic…
QCoreApplic…
QCoreApplic…
postEventSo…
g_main_cont…
?? [libglib-
g_main_cont…
QEventDispa…
QEventLoop:…
QEventLoop:…
QThread::ex…
QQmlThread…
QThreadPrivate::start(void*)
start_thread
__clone3
QSGBa…
QSGRen…
QSGRen…
QSGDef…
QQuick…
QSGRenderT…
QSGRenderTh…
QtPrivate
QtPrivate
QtPrivate
void doAc.
QMetaObj…
Quotient:
Quotient::
QtPrivate::QS
void doActiva
QMetaObject::
QNetworkRep…
QNetworkRep…
QNetworkRep…
QNetworkRep…
QMetaCallEvent::plac
QObject::event(QEven
QApplicationPrivate::r
QApplication::notify((
QCoreApplication::noti
QCoreApplication::senc
QCoreApplicationPrivat
QCoreApplication::senc
postEventSourceDispatc
g_main_context_dispatch
?? [libglib-2.0.so.0.7600.3]
g_main_context_iteration
QEventDispatcherGlib::processEvents(QFla
QXcbGlibEventDispatcher::processEvents(Q
QEventLoop::processEvents(QFlags<QEventL
QEventLoop::exec(QFlags<QEventLoop::Proc
QCoreApplication::exec()
QGuiApplication::exec()
QApplication::exec()
main
__libc_start_call_main
__libc_start_main_impl
_start
QGuiAp…
QWind…
xcbSour…
QCore…
QGuiA…
QAppli…
QAppli…
QObje…
QObje…
QWind…
QQuick…
QQuick…
QQuickA…
QQmlPri…
QQmlPri…
void qDel
QQmlAppl…
QQmlAppl…
QQmlAppl…
QQmlO…
QQmlC…
QQmlC…
QQmlC…
QQmlCom…
QQmlAp…
QQmlApplica…
Reserv…
__gnu_…
__gnu_…
std::__
std::__
std::__i
std::__sc
std::sort
Reservatio.
9.206E+09 aggregated cycles:u cost in total

# perf Timeline



# Powertop

```
PowerTOP 2.14     Overview   Idle stats   Frequency stats   Device stats   Tunables   WakeUp

Summary: 337.2 wakeups/second,  0.0 GPU ops/seconds, 0.0 VFS ops/sec and 16.8% CPU use

            Usage       Events/s    Category       Description
          1.6 ms/s      163.3       Timer          tick_sched_timer
         38.2 ms/s       14.2       Process        [PID 2213] /k/kde5/inst/bin/plasmashell --no-respa
         17.9 ms/s       18.1       Process        [PID 1267] /usr/bin/Xorg.bin -nolisten tcp -auth /
          0.9 ms/s       17.4       Interrupt      [30] radeon
         18.6 ms/s        4.3       Process        [PID 2184] /k/kde5/inst/bin/kwin_x11 --replace
        157.9 us/s       11.3       Process        [PID 15] [rcu_preempt]
          0.7 ms/s       11.0       Interrupt      [0] HI_SOFTIRQ
         11.9 ms/s        6.1       Process        [PID 2388] /k/kde5/inst/bin/ksystemstats
          0.9 ms/s        9.8       Process        [PID 10733] /usr/lib64/libreoffice/program/soffice
        208.4 us/s        9.8       Process        [PID 10386] /usr/lib64/firefox/firefox --sm-client
         18.7 ms/s        0.5       Process        [PID 9617] /usr/lib64/firefox/firefox -contentproc
        191.2 us/s        6.4       Interrupt      [3] net_rx(softirq)
          9.1 ms/s        1.2       Process        [PID 2467] /usr/lib64/firefox/firefox --sm-client-
          9.4 ms/s        0.4       Process        [PID 2424] /k/kde5/inst/bin/konsole -session 10ab6
          7.2 us/s        4.1       Timer          napi_watchdog
         81.4 us/s        4.0       Process        [PID 1193] containerd --config /var/run/docker/con
        110.9 us/s        3.4       Process        [PID 5664] /usr/sbin/mysqld --defaults-file=/home/
          4.7 ms/s        1.4       Process        [PID 11250] /usr/lib64/firefox/firefox -contentpro
         16.0 us/s        2.4       Interrupt      [6] tasklet(softirq)
          3.0 ms/s        1.2       Process        [PID 10525] /k/kde5/inst/bin/kontact
          1.0 ms/s        1.9       Process        [PID 1003] /usr/sbin/NetworkManager --no-daemon
         40.2 us/s        2.1       Interrupt      [4] block(softirq)
          1.2 ms/s        1.6       Process        [PID 2358] /usr/bin/nextcloud
        124.0 us/s        2.0       Process        [PID 1218] containerd --config /var/run/docker/con
         16.3 us/s        2.0       Timer          watchdog_timer_fn
         22.2 us/s        1.9       Process        [PID 76] [kcompactd0]
          3.9 ms/s       0.25       Process        [PID 2193] /k/kde5/inst/bin/kwin_x11 --replace
```

# Optimizing software

- Doing less is always good
  - Rare to find such cases though
- Tradeoffs
  - Doing the same thing faster
  - Doing the same thing using different types of hardware
  - Can be hard to predict, can depend on specific hardware

# Evaluating tradeoffs

- Few established guidelines for energy consumption
- Non-linear behavior vs intuition
- Measure and compare!

# Measuring energy consumption

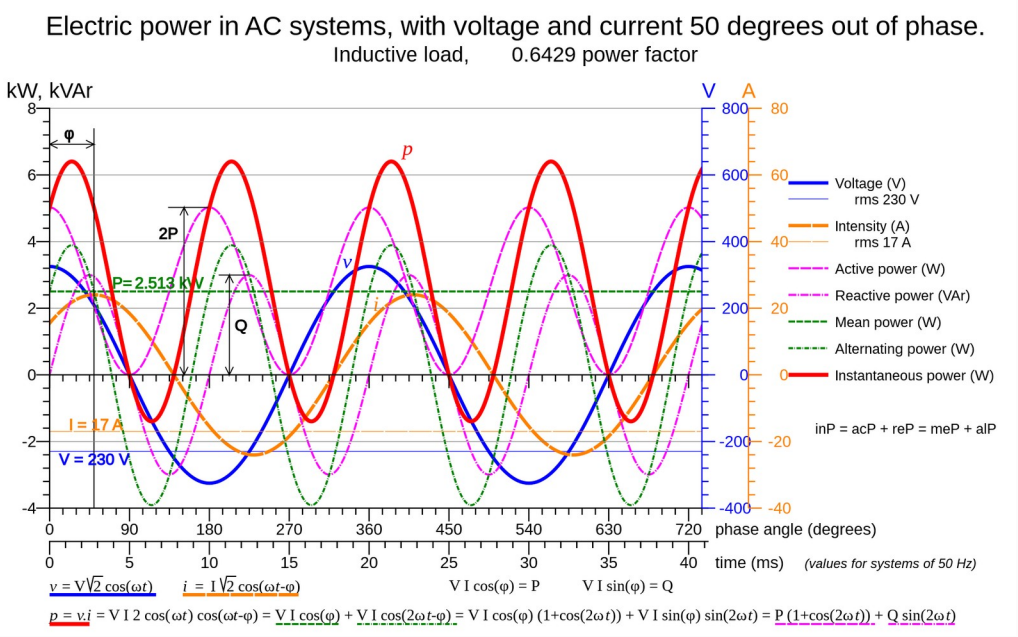
- Primary side (AC)
  - (+) complete (-) tricky for battery powered devices, safety
- Secondary side (DC)
  - (+) safety (o) misses power supply loss (-) difficult to access
- Built-in sensors
  - (+) availability, resolution (-) partial view, in-situ measurement

# Physics – The Good - DC

- Power (P), measured in Watt
  - P = Voltage (U) x Current (I)
- Energy: power over time
  - Measured in Ws, kWh or J/MJ

# Physics – The Bad - AC

- Power is now a complex number
  - Active power/real power (P), measured in Watt
  - Reactive power (Q), in VAr
  - Complex power (S), in VA
  - Apparent power (|S|), in VA
  - Power factor (α): P/Q



# Physics – The Ugly – Switched-mode Power Supplies

- Operating at ~100kHz - 2MHz
- Distort the AC sine wave
- Highly dynamic loads
- Results in noisy measurements when using a lower sampling rate

# AC Measurements – Household/IT devices

- Standalone devices: ~10€, no machine readable output
- IT server management devices: 300+€, Ethernet access
- Sample rate: <= 1Hz


# AC Measurements – Tasmota-based Wi-Fi plugs

- Switchable Wi-Fi power plugs with power meter
- ~8-20€ a piece
- Tasmota firmware (FOSS)
- Up to 5Hz sample rate
- Potential for more with modified firmware

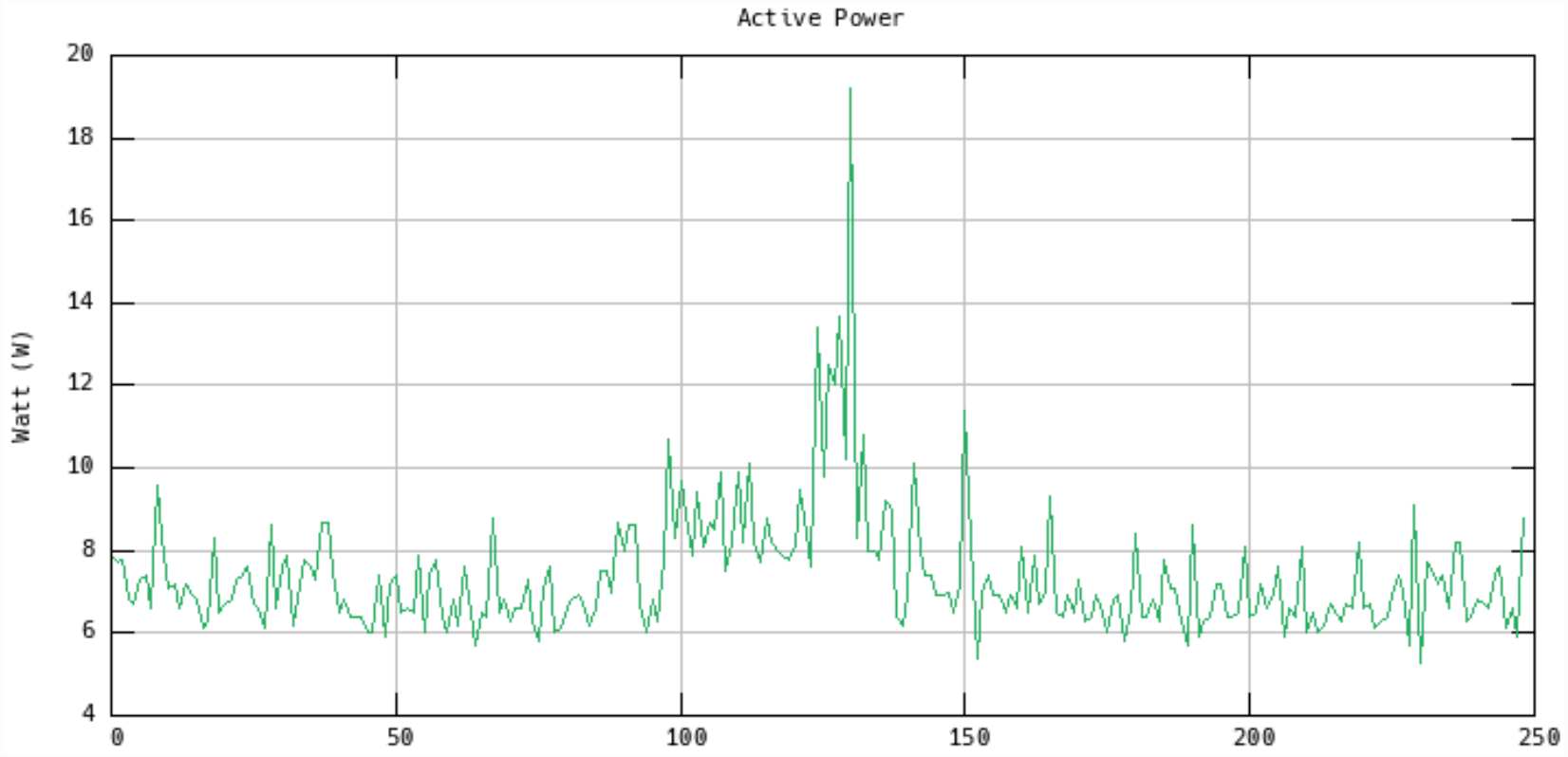
# AC Measurements – ALCIOM PowerSpy2

- Specialized device
- Serial over Bluetooth access
- Up to 50Hz sample rate
- Live data and recording
- ~300€
- Only partial protocol implementation

# AC Measurements - DIY

- Sensors only cost ~2$
- Development boards ~200-250$
- Sampling rate up to 1kHz
- Safety...

# AC Measurements - Example



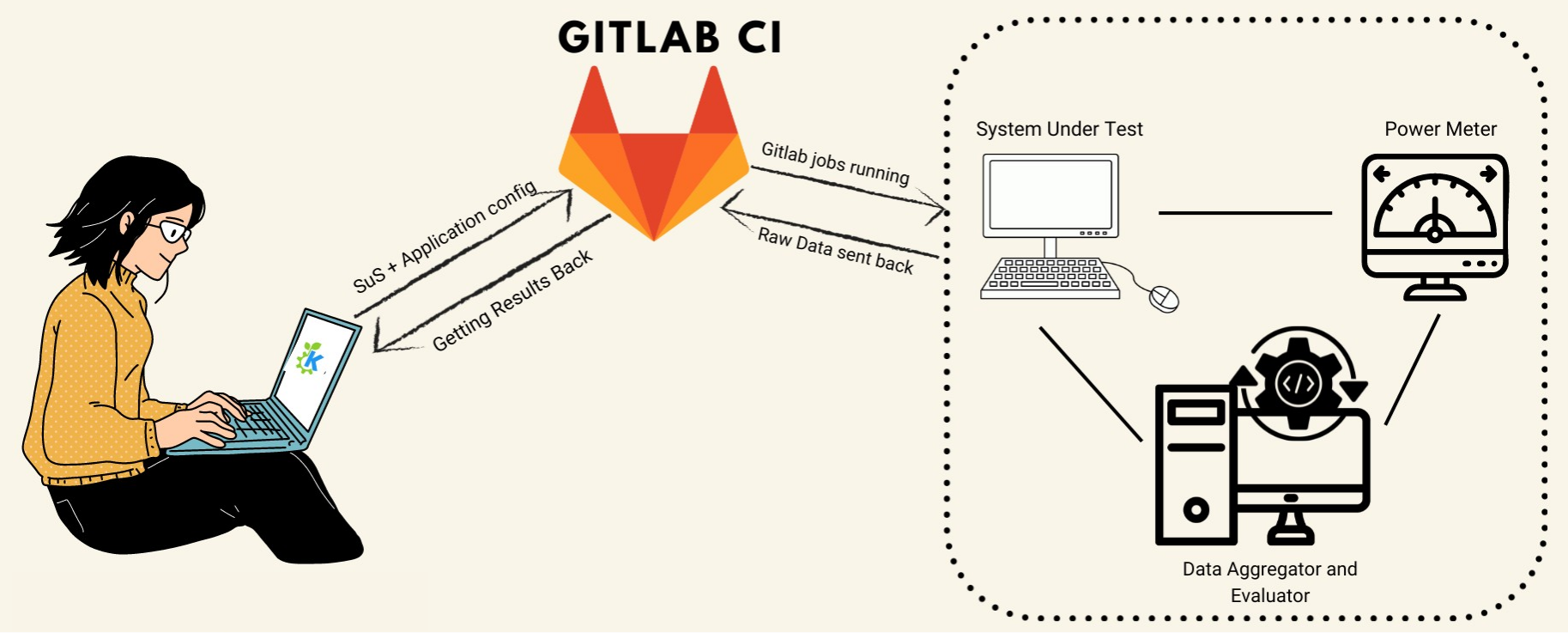
# Internal power sensors

- e.g. Intel RAPL
- Available sensors depend on hardware/hardware version
  - `perf list | grep energy`
- Data accessible via perf or specialized tools
  - `perf stat -e "power/energy-pkg/,power/energy-cores/" <app>`
  - `pinpoint --header -c <app>`
- Needs root access

# Internal power sensors - Example

# There is no cloud, just other people's power meters...

- GSoC "KEcoLab" by Karanjot Singh



# What do we do with all that?

- Do general profiling and optimizing first
- Focus on long-running processes
  - Continuous/frequent workloads
  - Idle behavior
- Research
  - Tooling and best practices are still very basic

# Thank you!

## Questions?

BoF

Monday 10:00 Room 1

# References

- https://volkerkrause.eu/2020/10/17/kde-cheap-power-measurement-tools.html
- https://tasmota.github.io/docs
- https://invent.kde.org/vkrause/powerspy2-tools
- https://github.com/osmhpi/pinpoint/
- https://eco.kde.org/blog/2023-06-13-gsoc23-energy-measurement-lab/