

Energy-Aware Computing Systems (EASY)

Energy measurement

2019-05-03

Timo Hönig, Stefan Reif, Benedict Herzog

Lehrstuhl für Informatik 4
Friedrich-Alexander-Universität Erlangen-Nürnberg



Lehrstuhl für Verteilte Systeme
und Betriebssysteme



FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

Energy Measurement

- Physics: $E_{t_0}(t) = \int_{t_0}^t P(\tau) d\tau = \int_{t_0}^t V(\tau) I(\tau) d\tau$

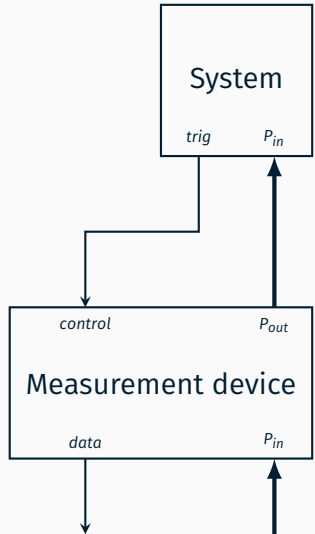
Energy Measurement

- Physics: $E_{t_0}(t) = \int_{t_0}^t P(\tau) d\tau = \int_{t_0}^t V(\tau) I(\tau) d\tau$
- CMOS power model: $P = \underbrace{\alpha C f V^2}_{P_{dynamic}} + \underbrace{V I_{leak}}_{P_{static}}$

Energy Measurement

- Physics: $E_{t_0}(t) = \int_{t_0}^t P(\tau) d\tau = \int_{t_0}^t V(\tau) I(\tau) d\tau$
- CMOS power model: $P = \underbrace{\alpha C f V^2}_{P_{dynamic}} + \underbrace{V I_{leak}}_{P_{static}}$
- How to measure the energy demand?

Practical Energy Measurement

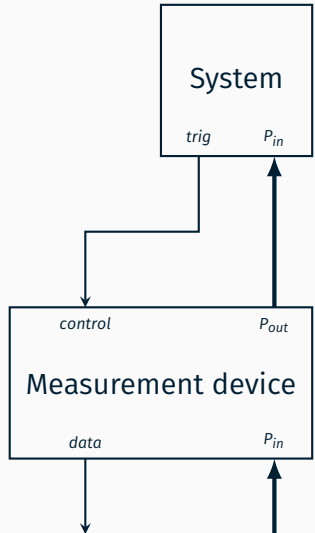


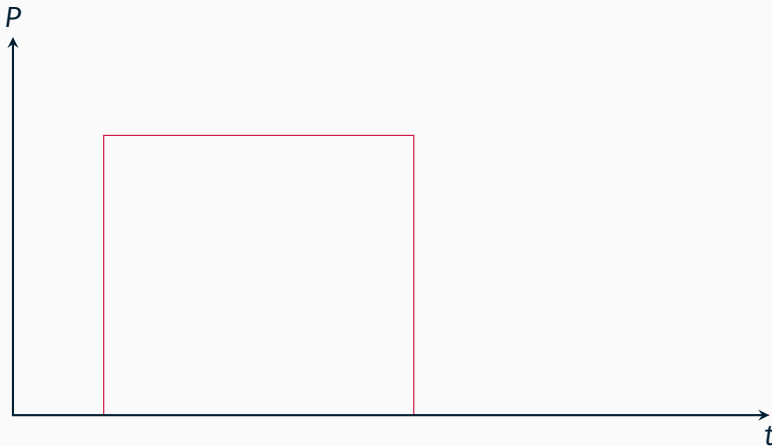
Practical Energy Measurement

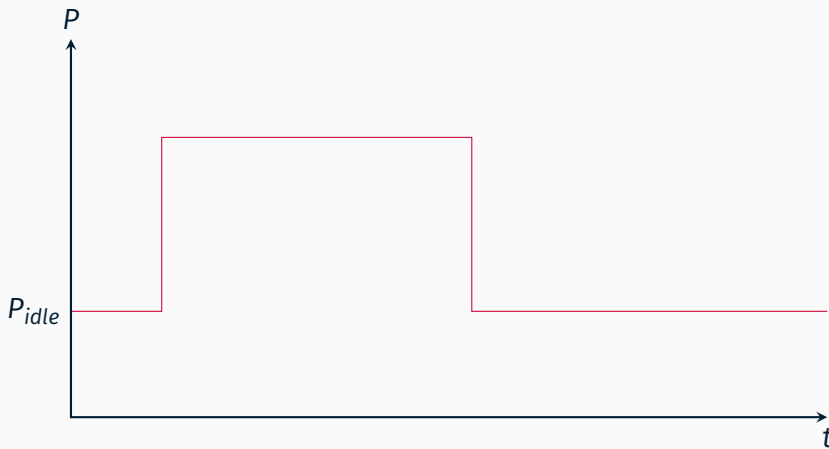
- Dedicated measurement device monitors power supply
- Discrete & periodic power sampling and aggregation

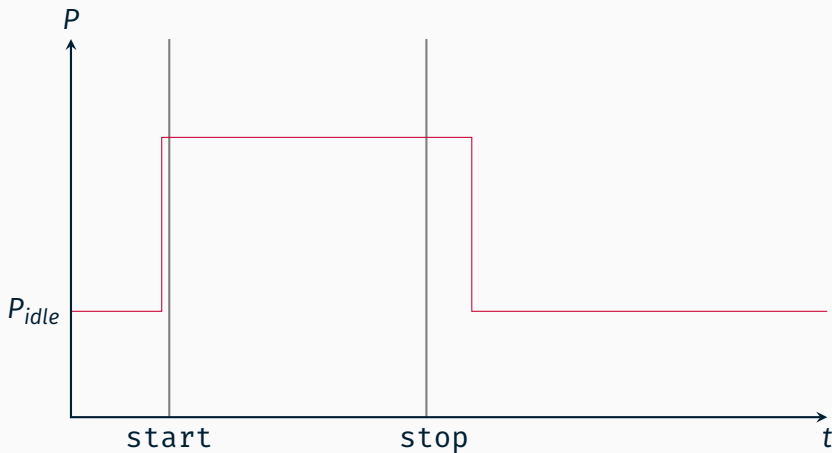
Typical usage

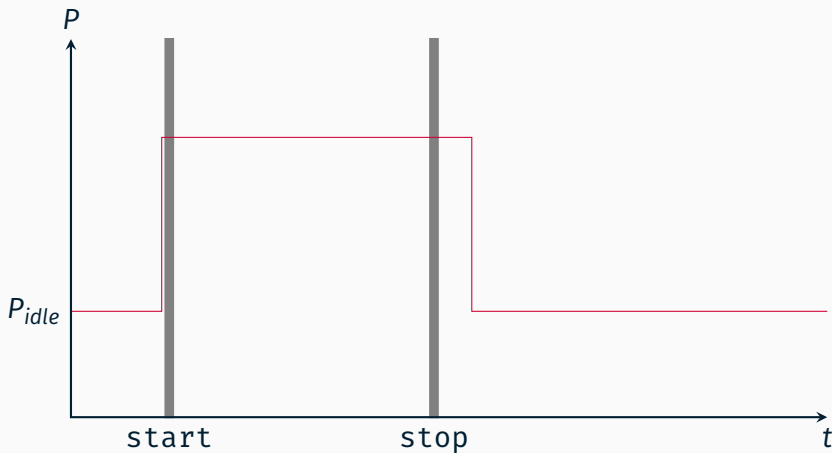
```
01 E1 = read();  
02 foo();  
03 E2 = read();  
04  $\Delta E = E2 - E1;$ 
```

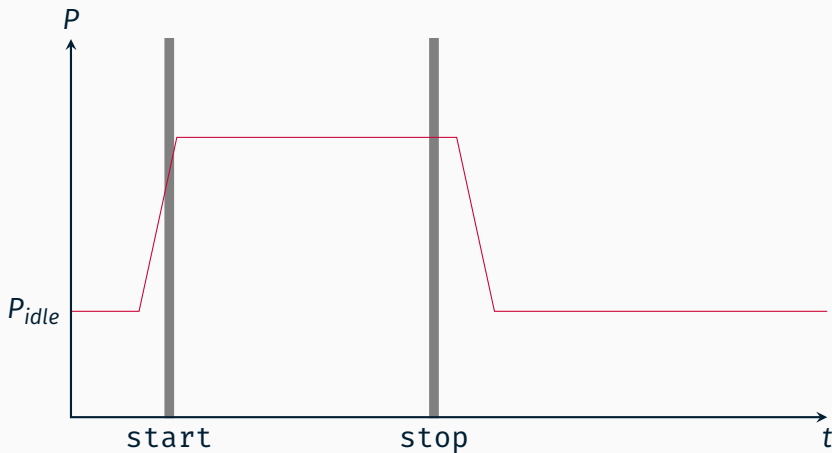


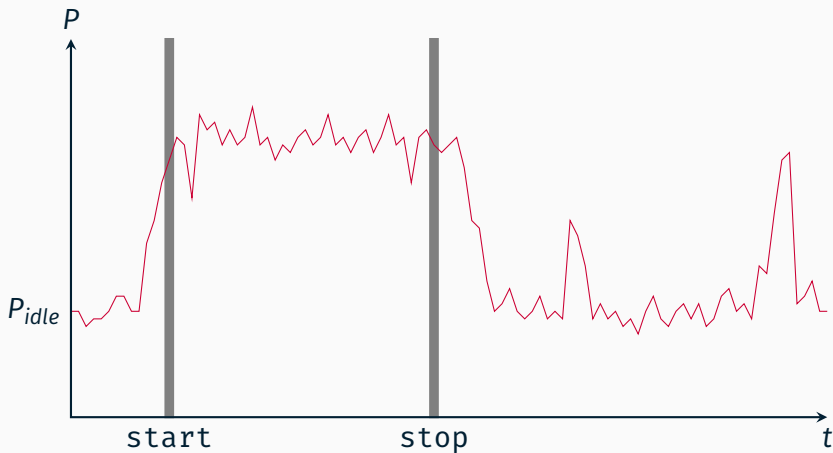












Energy measurement “devices”

- Running average power Limit (RAPL):
 - Built-in interface for power/energy measurement and configuration
 - x86-specific
- Domains:
 - package** processor package
 - ppo** processor core(s)
 - pp1** processor uncore
 - dram** memory (potentially unsupported)
 - psys** platform

- Internal calculation of power-over-time integral:

```
01 e1 = rapl_read();  
02 foo();  
03 e2 = rapl_read();  
04 return e2 - e1;
```

- Source code is available online:

→ [http://web.eece.maine.edu/~vweaver/
projects/rapl/rapl-read.c](http://web.eece.maine.edu/~vweaver/projects/rapl/rapl-read.c)

- Microchip MCP39F511N (“easypower”)
 - Monitoring of 230V power supply
 - Output: (timestamp, power) tuples
 - Driver in /proj/i4easy/pub/

Foobar

```
01 ...
02 1553071638.93154 s    16.97 W    0.05 W
03 1553071638.93363 s    17.50 W    0.11 W
04 ...
```

- Microchip MCP39F511N (“easypower”)
 - Monitoring of 230V power supply
 - Output: (timestamp, power) tuples
 - Driver in /proj/i4easy/pub/

Foobar

01 ...

02 1553071638.93154 s

16.97 W

0.05 W

03 1553071638.93363 s

17.50 W

0.11 W

04 ...

Timestamp

Ch 1

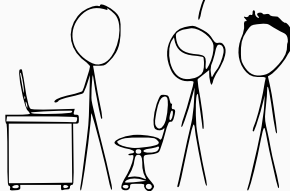
Ch 2

git

THIS IS GIT. IT TRACKS COLLABORATIVE WORK
ON PROJECTS THROUGH A BEAUTIFUL
DISTRIBUTED GRAPH THEORY TREE MODEL.

COOL. HOW DO WE USE IT?

NO IDEA. JUST MEMORIZE THESE SHELL
COMMANDS AND TYPE THEM TO SYNC UP.
IF YOU GET ERRORS, SAVE YOUR WORK
ELSEWHERE, DELETE THE PROJECT,
AND DOWNLOAD A FRESH COPY.



Create a repository

```
01 user:~$ mkdir example
02 user:~$ cd example
03 user:~/example$ git init
04 Initialized empty Git repository in /example/.git/
```

Add files

```
01 user:~/example$ touch README.md
02 user:~/example$ git add README.md
03 user:~/example$ git commit -m "initialer commit"
04 [master (root-commit) 16e1b9b] initial commit
05 1 file changed, 0 insertions(+), 0 deletions(-)
06 create mode 100644 README.md
```

16e1b9b



Track changes

```
01 user:~/example$ echo "23" > foo
02 user:~/example$ git add foo
03 user:~/example$ git commit -m "set foo to 23"
04 [master 3c49a82] set foo to 23
05 1 file changed, 1 insertion(+)
06 create mode 100644 foo
```

16e1b9b 3c49a82



```
01 user:~/example$ echo "42" > foo
02 user:~/example$ git add foo
03 user:~/example$ git commit -m "another commit"
04 [master 26d0c4d] another commit
05 1 file changed, 1 insertion(+), 1 deletion(-)
```



```
01 user:~/example$ echo "1337" > bar
02 user:~/example$ echo "zero" > foo
03 user:~/example$ git add bar foo
04 user:~/example$ git commit -m "another change"
05 [master a9a63dc] another change
06 2 files changed, 2 insertions(+), 1 deletion(-)
07 create mode 100644 bar
```



Show status

```
01 user:~/example$ git status
02 On branch master
03 nothing to commit, working tree clean
```



master

Create a new branch

```
01 user:~/example$ git checkout -b foobaz 3c49a82
02 Switched to a new branch 'foobaz'
03 user:~/example$ git shortlog
04 Demo User (2):
05   initial commit
06   set foo to 23
```

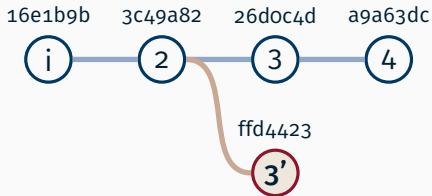


master

foobaz

Make changes on new branch

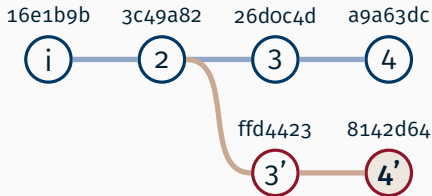
```
01 user:~/example$ echo "3.141" > baz
02 user:~/example$ git add .
03 user:~/example$ git commit -m "new file"
04 [foobaz ffd4423] new file
05 1 file changed, 1 insertion(+)
06 create mode 100644 baz
```



Make changes on new branch

(2)

```
01 user:~/example$ echo "pi" > baz
02 user:~/example$ git commit -am "message"
03 [foobaz 8142d64] message
04 1 file changed, 1 insertion(+), 1 deletion(-)
```

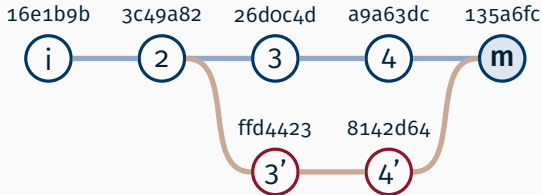


master

foobaz

Merge branches

```
01 user:~/example$ git checkout master
02 user:~/example$ git merge foobaz -m "merge commit"
03 Merge made by the 'recursive' strategy.
04   baz | 1 +
05   1 file changed, 1 insertion(+)
06   create mode 100644 baz
```

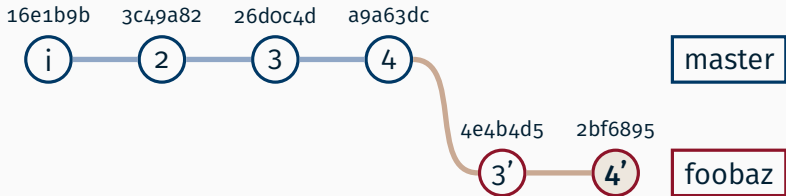


master

foobaz

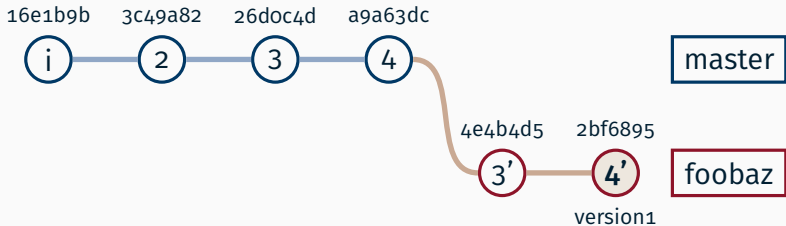
Alternatively: rewrite history

```
01 user:~/example$ git rebase master
02 First, rewinding head to replay your work on top of
   it...
03 Applying: new file
04 Applying: message
```



Create tags

```
01 user:~/example$ git tag version1
```



Cheatsheet

git init Create new repository in the current directory

git add *file* Mark file as candidate for next *commit*

git commit Record changes

git diff Show changes

git show Show most recent change

git status Show unrecorded changes

git branch Show available branches

git log Show history

man *git-command* Show help, e.g. man git-add

Cheatsheet (remote repositories)

git clone *URL* Initial copy from a remote repository

git fetch *name* Download changes from remote repository

git pull *name* Shortcut for fetch and merge

git checkout *branch* Change current branch

git remote add *URL* Add remote repositories

git push *name* Upload changes to remote repository

git push --tags Upload tags to remote repository