

What if removing software features to make programs smaller doesn't save energy... but sometimes actually uses more?

# On the Effect of Feature Reduction on Energy Consumption: An Exploratory Study

Xhevahire Tërnav<sup>1</sup>, Romain Lefeuvre<sup>2</sup>, Quentin Perez<sup>2,3</sup>,  
Djamel Eddine Khelladi<sup>2,4</sup>, Mathieu Acher<sup>2,5</sup>, Benoit Combemale<sup>2</sup>

<sup>1</sup>LTCl, Télécom Paris, Institut Polytechnique de Paris

<sup>2</sup>Univ Rennes, Inria, IRISA, <sup>3</sup>INSA Rennes, <sup>4</sup>CNRS, <sup>5</sup>IUF  
Paris - Rennes, France

SPLC'25, September 01-05, 2025

# Context

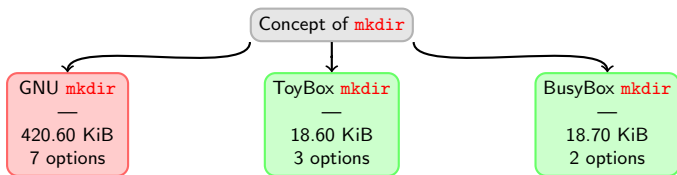
- Software powers everything in our digital society. But ICT also consume a significant amount of energy, raising environmental concerns
  - Configurable systems let developers enable or disable features, but over time they become bloated and complex
  - Traditionally, 'debloating' is meant to:
    - ⊗ Shrink binaries
    - ⊗ Reduce attack surfaces
    - ⊗ Improve performance
- 
- But our starting point was:
    - ⊗ Does feature reduction = energy reduction?

# Research gap

- Prior work has explored energy in **configurable systems** (e.g., feature interactions, static analysis, execution time)
- **Feature reduction** for e.g., reducing attack surface is studied
- But to our knowledge, no previous work has studied **their** combined effect on **energy consumption**

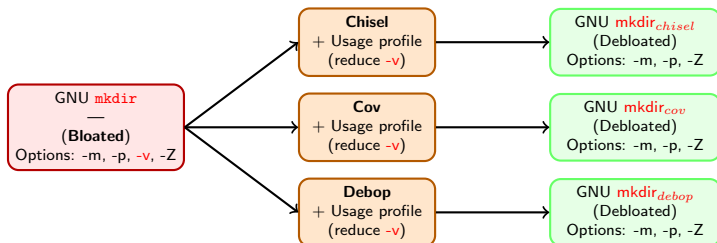
# Two types of feature reduction

**Built-in reduction:** Developers intentionally create alternative implementations with fewer features



# Two types of feature reduction

**On-demand reduction:** Developers (build tools, and) debloat software to remove unnecessary features




# Research questions

- I. For **built-in reduction**, we ask how three factors:

$RQ_{1.1}$  : binary size

$RQ_{1.2}$  : # configuration options

$RQ_{1.3}$  : execution time


impact  energy consumption

- II. For **on-demand**, we asked the same with two factors:

$RQ_{2.1}$  : binary size

- : ~~# configuration options~~

$RQ_{2.2}$  : execution time

impact  energy consumption

- Then, we wondered about their  correlations

# Methodology for built-in reduction

- Q 28/75 programs (e.g., mv, ls, mkdir) compared their alternative implementations across GNU, ToyBox, and BusyBox
- Q For each program: 2 common valid configurations + input

---

- Measured: binary size, # configuration options, execution time, and energy consumption

- 
- Energy consumption is measured using the **Jouleit**<sup>1</sup>, which leverages Intel's RAPL counters
  - Each measurement was repeated 10 times (1,680, in total) on a controlled hardware and OS setup, following best practices to minimize noise

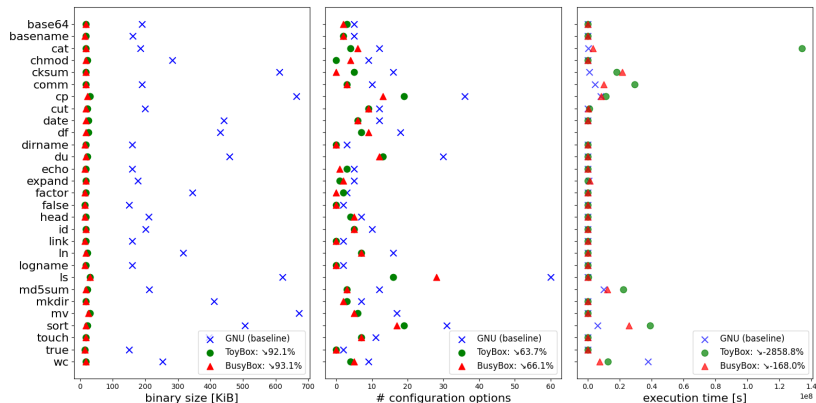
---

<sup>1</sup> <https://github.com/powerapi-ng/jouleit>

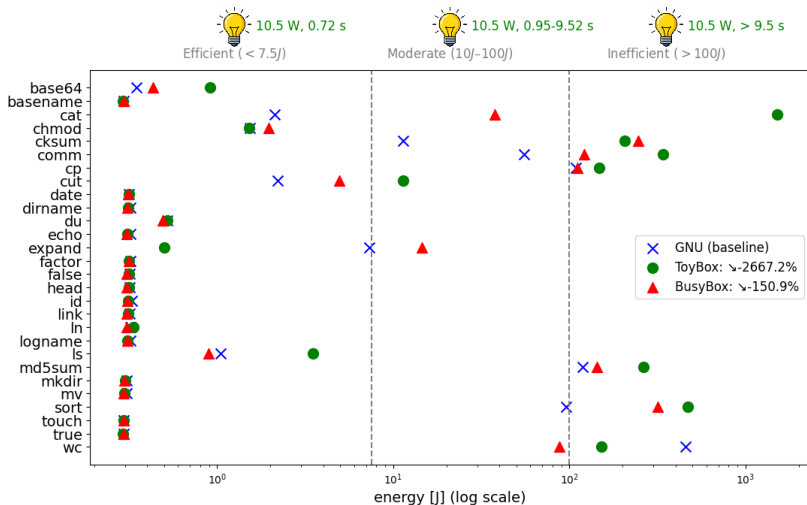


# Built-in: Binary size, options, and runtime *reduction*

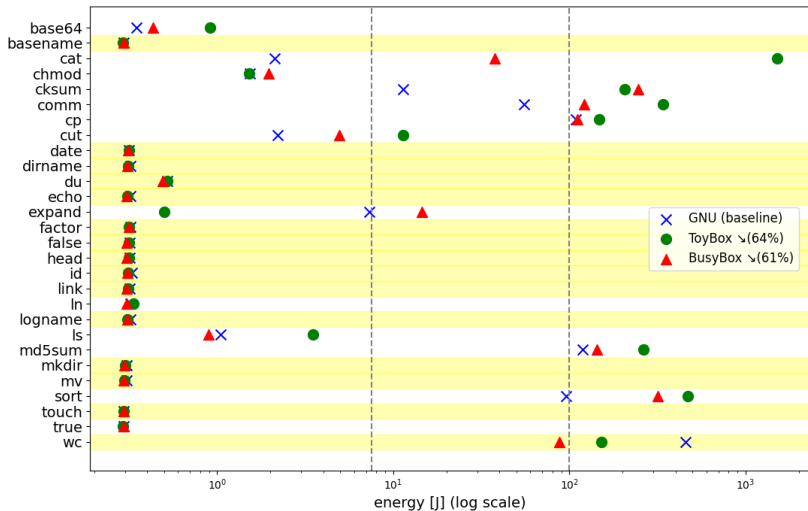
- Drastically smaller binaries: all, 92.1% and 93.1%
- Far fewer configuration options: all, 63.7% and 66.1%
- Execution time varied per program: (few outliers)



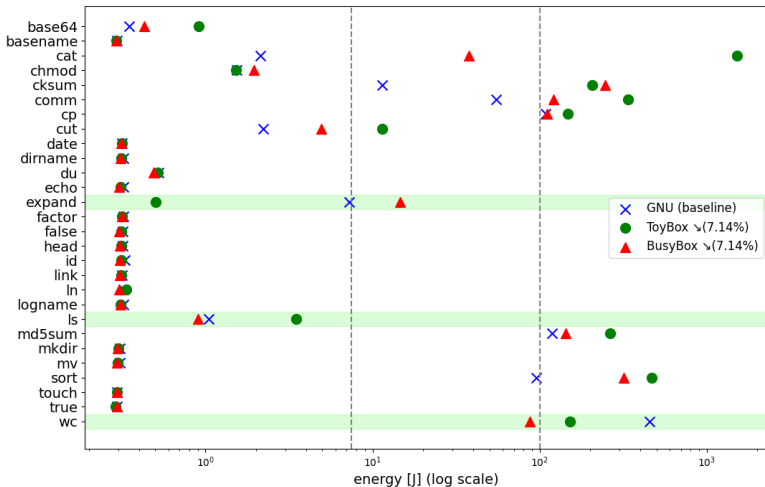
# Built-in: Energy consumption per program



# Built-in: Comparative analysis (lower consumption)

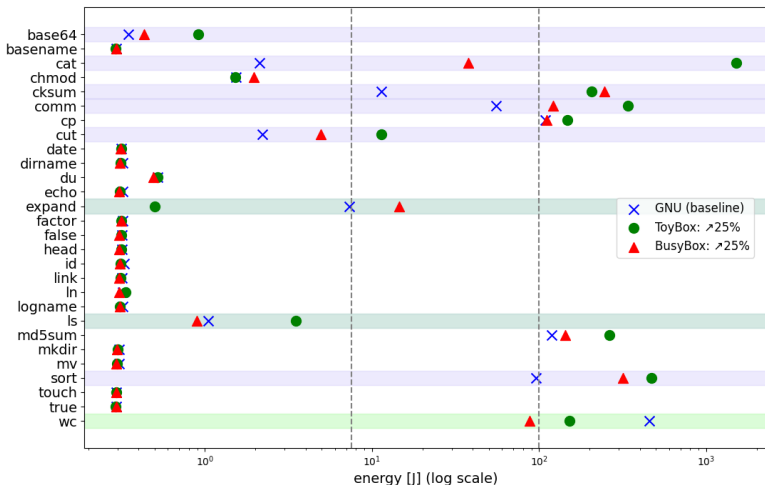


# Built-in: Comparative analysis (lower and significant)

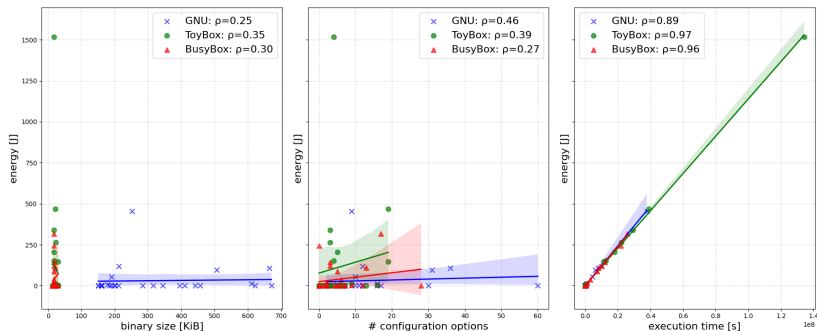


# Built-in: Comparative analysis (higher and significant)

- 36% of ToyBox and 39% of BusyBox programs use more energy than GNU versions. Whereas, with  $p > 0.05$  are:



# Built-in: Energy correlates with execution time



**Key insight:** Energy consumption is much more strongly tied to execution time ( $RQ_{1.3}$ ) than to binary size ( $RQ_{1.1}$ ) or number of options ( $RQ_{1.2}$ ).

# Methodology for on-demand reduction

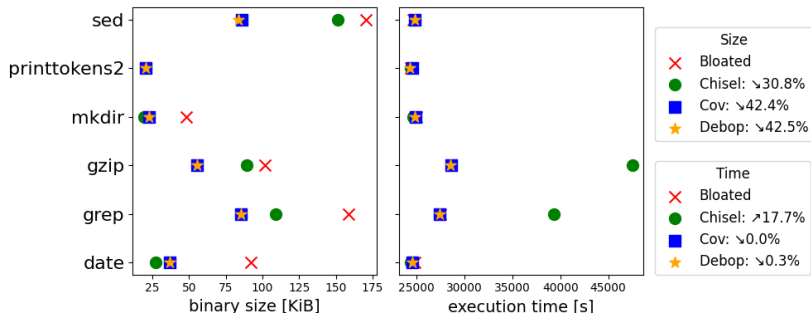
- ✔ 6 GNU programs debloated with Chisel, Debop, and Cov<sup>2</sup>
  - ✂ Source level debloating of a program, regarding runtime features + 2 usage profiles → controlled, on-demand reduction
    - The selected programs are the same as in the first experiment
- 
- Measured: binary size, execution time, and energy consumption
- 
- Energy consumption is measured using the **Jouleit**
  - Each measurement was repeated 10 times (480, in total), following the same methodology as in the first experiments

---

<sup>2</sup>Qi Xin, et.al. Studying and Understanding the Tradeoffs Between Generality and Reduction in Software Debloating. ASE'22. <https://doi.org/10.1145/3551349.3556970>

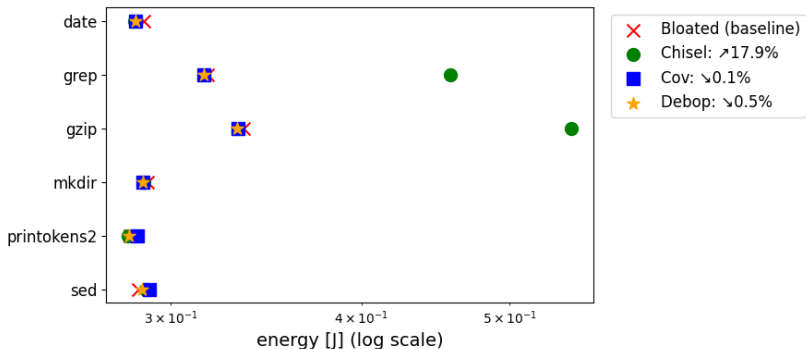
# On-demand: Binary size and execution time *reduction*

- Smaller binaries: all, 30.8%, 42.4%, and 42.5%
- Execution time varied per program: 17.7% slower, 0%, 0.3% faster



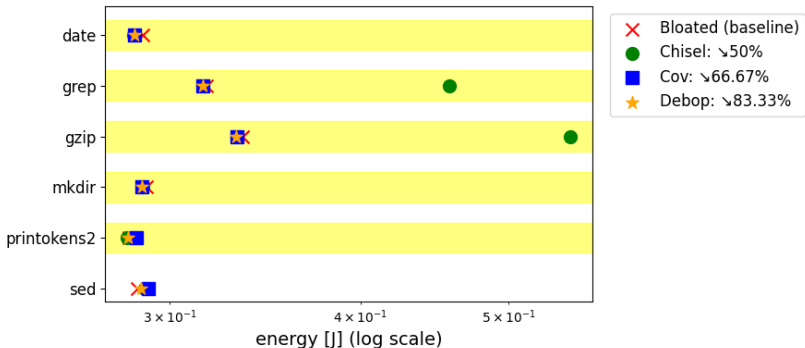


# On-demand: Energy consumption per program

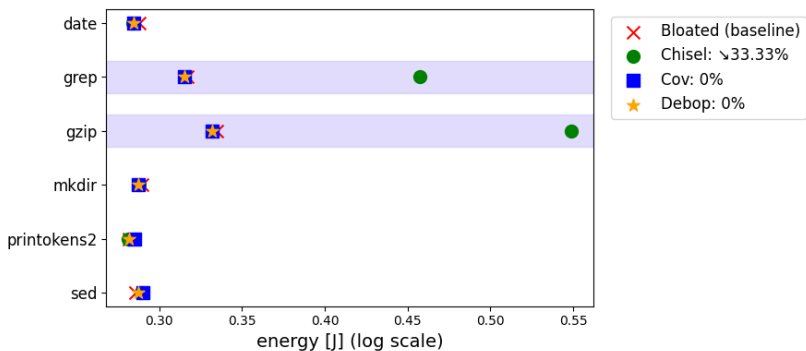


# Comparative analysis (lower and significant)

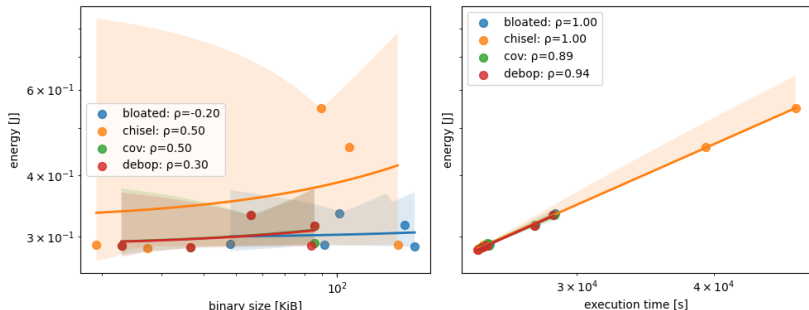
- No significant cases among programs that consumed less



# Comparative analysis (higher and significant)



# On-demand: Energy correlates with execution time



**Key insight:** Energy consumption ( $RQ_{2.2}$ ) is strongly tied to execution time, while reducing binary size through debloating ( $RQ_{2.1}$ ) does not consistently lower energy use and can even increase it.

# Unintended energy impacts of feature reduction

- The impact of feature reduction on energy consumption can be counterintuitive
- **In-depth analysis:** debloating removed optimizing code, leading to higher energy consumption despite fewer executed lines

```
1 grep-2.4.2 - prtext()
2 // ...
3 if (!out_quiet) {
4     if (pending > 0) {
5         prpending(beg);
6     }
7 // ...
```

```
1 gzip-1.3 - ct_init()
2 // ...
3 if ((int) static_dtree[0].dl.len != 0) {
4     return;
5 }
6 // ...
```

- **Mono-objective debloating** (e.g., reducing binary size or attack surface) can harm other properties, including energy consumption



# Energy Consumption Insights from Feature Reduction

- First exploratory study on the impact of feature reduction on energy consumption
- We distinguished built-in vs. on-demand reduction
- Does removing features always save energy?
- Not necessarily, but smarter, energy-aware *debloating* can

- 
- Replication package:

[swh:1:rev:85286751845d59e9d032ffc0b91b92b2220954df](https://sw.hawaii.edu/~mccall/sw/1:rev:85286751845d59e9d032ffc0b91b92b2220954df)