Incremental Exploration of Linux Configuration Space

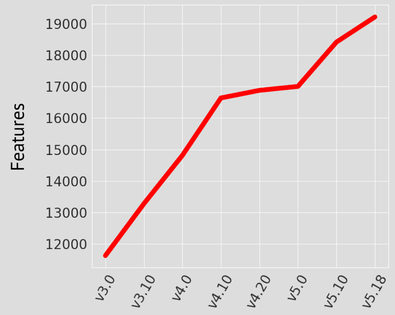
Georges Aaron Randrianaina[1,4,5], Djamel Eddine Khelladi[2,4,5], Olivier Zendra[3,4,5], Mathieu Acher[1,4,5,6]

Univ Rennes[1], CNRS[2], Inria[3], IRISA[4], DiverSE[5], INSA Rennes[6], IUF[7]

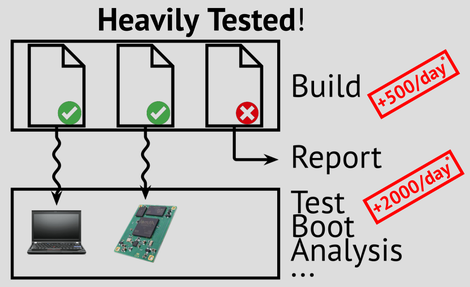
Context and Motivation

Case Study:
The Linux Kernel

A **Highly-configurable** and **Fast Moving System**



That **Needs** to be **Heavily Tested**!



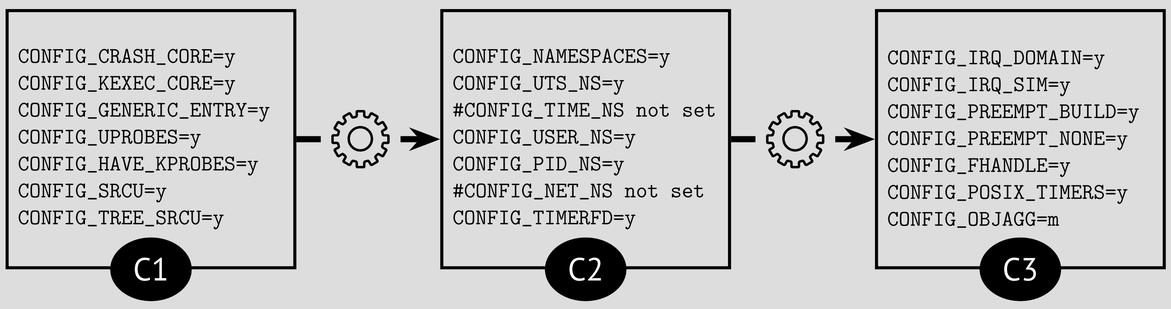
* For x86_64 not counting other platforms (RISC-V, ARM, MIPS,...)

*"The Linux Kernel does not have a test suite[...] for hardware interaction [...] The best thing you can ever do for us is: you **just build the Kernel** and tell us if you have a problem. That is our QA cycle."*
–Greg Kroah-Hartman, Linux maintainer

Challenge and Objectives

- Building a x86_64 configuration can take up to 10 min so it would take about **a week of CPU-time for 1000** of them!
- We aim to **reduce the build time** by **reusing the artefact** of previous builds: this technique is called **Incremental Build**
- Incremental build has been studied on code change
- Our contribution is to **Bring Incremental Build to Software Configurations Changes**[1, 2, 3]

Vision

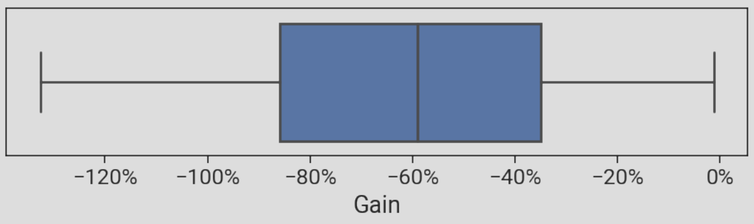


25min VS 30min
Incremental Clean

Naive incremental approach

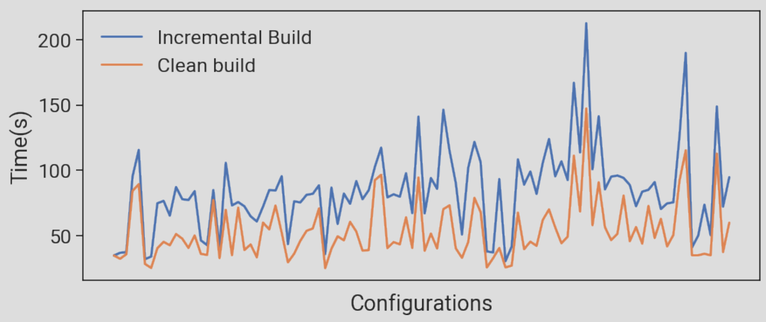
Using Make's incremental strategy gives negative gains

- We rely on the incremental strategy of Make that captures the dependencies to decide which parts to rebuild
- We built 50 random configurations in a row without running `make clean`



Using a Compiler Cache (Ccache) adds an overhead

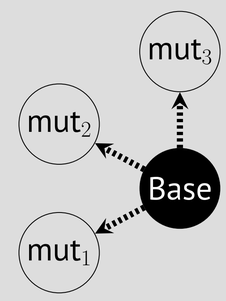
- *Ccache*[4] is a compiler cache that speeds up recompilation by caching previous compilations and detecting when the same compilation is being done again
- We built 100 random configurations in a row using *Ccache*



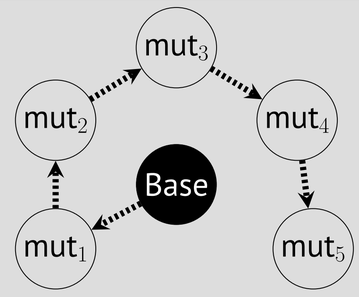
References
[1] Georges Aaron Randrianaina, Djamel Eddine Khelladi, Olivier Zendra, and Mathieu Acher. Towards Incremental Build of Software Configurations. ICSE'22, NIER
[2] Georges Aaron Randrianaina, Xhevahire Tërnava, Djamel Eddine Khelladi, and Mathieu Acher. On the Benefits and Limits of Incremental Build of Software Configurations: An Exploratory Study. ICSE'22
[3] Georges Aaron Randrianaina. Incremental Build of Linux Kernel Configurations. In EuroDW'22
[5] ccache.dev

Our New PyroBuildS Approach

Exploration strategies



Radial Exploration



Snake Exploration

Research questions

- RQ1 – *Correctness and consistency: Are* PyroBuildS *incremental builds correct and consistent with clean builds?*
- RQ2 – *Cost reduction: Are* PyroBuildS *incremental builds faster than clean builds?*
- RQ3 – *Diversity: Does* PyroBuildS *explore the configuration space for sufficient diversity?*

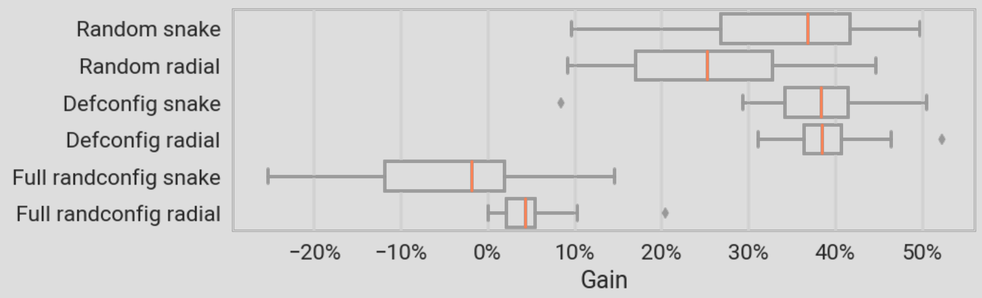
Preliminary results

RQ1 – Correctness and consistency

Base	Mode	Correctness (%)	Consistency (%)
Default	radial	100%	100%
	snake	100%	100%
Random	radial	80.5%	100%
	snake	94.5%	100%
	snake (Random)	85.2%	97.0%
	radial (Random)	85.4%	98%

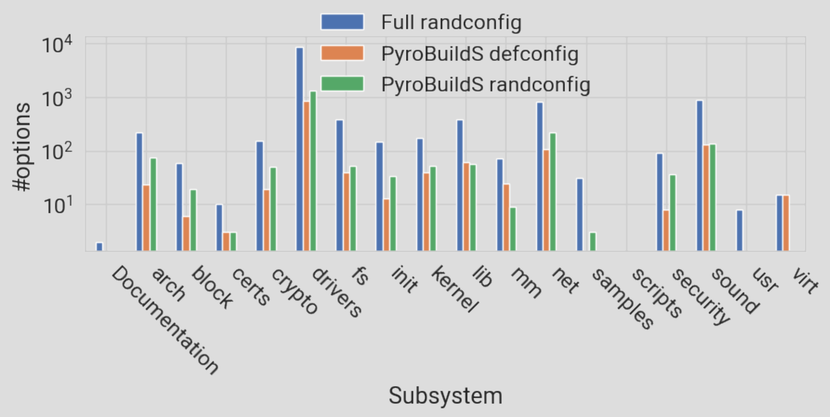
Incremental *PyroBuildS* mutations achieve 100% correctness with defconfig, 80% with random base for radial, and 94.5% for snake, all maintaining consistency compared to full randconfigs

RQ2 – Cost reduction



Using random configurations in radial and snake explorations, incremental builds experience losses of -79% and -288%, while *PyroBuildS* mutations, with both random and default configurations, limit losses to just -20%, with gains of over 80%

RQ3 – Diversity



PyroBuildS covers 14/18 of Linux's subsystems and 33% of the options covered by randconfigs.

We described PyroBuildS, our new approach to incrementally explore the (very large) configuration space of Linux, showing that appropriate exploration strategies trigger synergies with these caching capabilities of Make.
Overall, mutation-based builds (1) provide a tradeoff between diversity, build time, and correctness; (2) are an interesting complement to random configurations.

Made with Emacs, LaTeX and TikZ