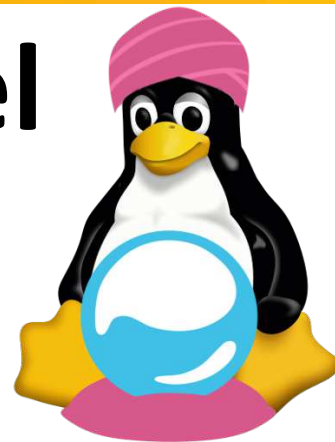




Embedded Linux  
Conference  
Europe

# Learning the Linux Kernel Configuration Space: Results and Challenges



Prof. Mathieu Acher (University of Rennes 1, Inria/IRISA)

@acherm



# Talk given at ELC 2019 (30 october 2019)

- Abstract: “Given a configuration, can humans know in advance the size, the compilation time, or the boot time of a Linux kernel? Owing to the huge complexity of Linux (there are more than 15000 options with hard constraints and subtle interactions), machines should rather assist contributors and integrators in mastering the configuration space of the kernel. In this talk, Mathieu Acher will introduce TuxML an OSS tool based on Docker/Python to massively gather data about thousands of kernel configurations. Mathieu will describe how 200K+ configurations have been automatically built and how machine learning can exploit this information to predict properties of unseen Linux configurations, with different use cases (identification of influential/buggy options, finding of small kernels, etc.) The vision is that a continuous understanding of the configuration space is undoubtedly beneficial for the Linux community, yet several technical challenges remain in terms of infrastructure and automation.”
- This research was funded by the ANR-17-CE25-0010-01 VaryVary project
  - <https://varyvary.github.io/>

- Learning From Thousands of Build Failures of Linux Kernel Configurations
  - Mathieu Acher, Hugo Martin, Juliana Alves Pereira, Arnaud Blouin, Djamel Eddine Khelladi, Jean-Marc Jézéquel
  - <https://hal.inria.fr/hal-02147012>
- Learning Very Large Configuration Spaces: What Matters for Linux Kernel Sizes
  - Mathieu Acher, Hugo Martin, Juliana Pereira, Arnaud Blouin, Jean-Marc Jézéquel, Djamel Eddine Khelladi, Luc Lesoil, Olivier Barais
  - <https://hal.inria.fr/hal-02314830>



# Linux everywhere since highly configurable

```
config X86_X2APIC
    bool "Support x2apic"
    depends on X86_LOCAL_APIC && X86_64 && (IRQ_REMAP || HYPERVISOR_GUEST)
    ---help---
        This enables x2apic support on CPUs that have this feature.

        This allows 32-bit apic IDs (so it can support very large systems),
```

```
config IOSF_MBI
    tristate "Intel SoC IOSF Sideband support for SoC platforms"
    depends on PCI
    ---help---
        This option enables sideband register access support for Intel SoC
        platforms. On these platforms the IOSF sideband is used in lieu of
        MSR's for some register accesses, mostly but not limited to thermal
        and power. Drivers may query the availability of this device to
        determine if they need the sideband in order to work on these
        platforms. The sideband is available on the following SoC products.
```

```
#
# Processor type and features
#
# CONFIG_ZONE_DMA is not set
# CONFIG_SMP is not set
# CONFIG_X86_FEATURE_NAMES is not set
# CONFIG_X86_FAST_FEATURE_TESTS is not set
CONFIG_X86_X2APIC=y
CONFIG_X86_MPPARSE=y
CONFIG_GOLDFISH=y
# CONFIG_INTEL_RDT_A is not set
# CONFIG_X86_EXTENDED_PLATFORM is not set
CONFIG_IOSF_MBI=m
CONFIG_IOSF_MBI_DEBUG=y
CONFIG_X86_SUPPORTS_MEMORY_FAILURE=y
# CONFIG_SCHED_OMIT_FRAME_POINTER is not set
```

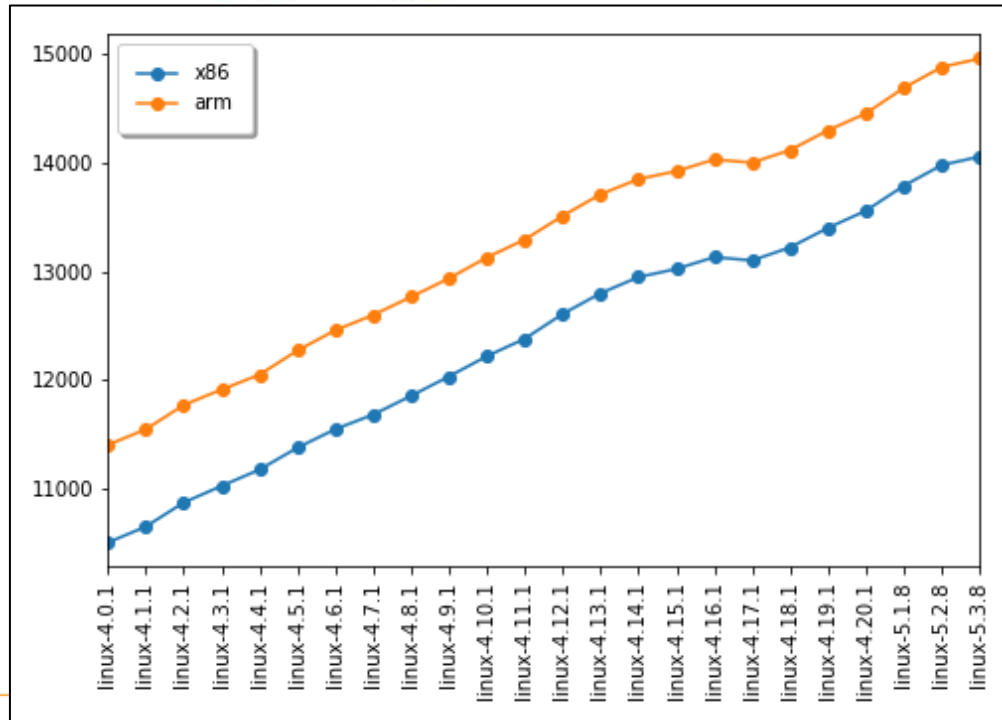
**Kconfig files/doc**

**.config**





# 15,000+ options



## Configurations: Hell or Heaven?

Stop ou encore?

# How to ensure that all configurations of the Linux kernel build/boot?

Many failures are due to buggy (combinations of) options  
Devs/maintainers are struggling to track/fix bugs  
Linus Torvalds: “random crazy user bugs” (random configurations are certainly a good subset)

**Testing** Linux kernels (on few configs):  
(e.g., 0-day/KernelCI)



# Given a kernel configuration, what's its size/boot time?

```
#  
# Processor type and features  
#  
# CONFIG_ZONE_DMA is not set  
# CONFIG_SMP is not set  
# CONFIG_X86_FEATURE_NAMES is not set  
# CONFIG_X86_FAST_FEATURE_TESTS is not set  
CONFIG_X86_X2APIC=y  
CONFIG_X86_MPPARSE=y  
CONFIG_GOLDFISH=y  
# CONFIG_INTEL_RDT_A is not set  
# CONFIG_X86_EXTENDED_PLATFORM is not set  
CONFIG_IOSF_MBI=m  
CONFIG_IOSF_MBI_DEBUG=y  
CONFIG_X86_SUPPORTS_MEMORY_FAILURE=y  
# CONFIG_SCHED_OMIT_FRAME_POINTER is not set
```



## Who knows what's the effect of options?

Default configurations/options' values

Documentation (Kconfig)

Configurators

# Effects of (combinations of) options on build status/boot/size/boot time/performance/security

```
# Processor type and features
#
# CONFIG_ZONE_DMA is not set
# CONFIG_SMP is not set
# CONFIG_X86_FEATURE_NAMES is not set
# CONFIG_X86_FAST_FEATURE_TESTS is not set
CONFIG_X86_X2APIC=y
CONFIG_X86_MPPARSE=y
CONFIG_GOLDFISH=y
# CONFIG_INTEL_RDT_A is not set
# CONFIG_X86_EXTENDED_PLATFORM is not set
CONFIG_IOSF_MBI=m
CONFIG_IOSF_MBI_DEBUG=y
CONFIG_X86_SUPPORTS_MEMORY_FAILURE=y
# CONFIG_SCHED_OMIT_FRAME_POINTER is not set
```

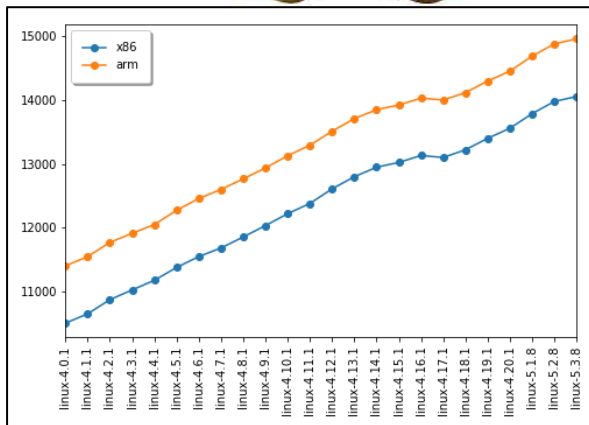


**General problem:**

**Taming the configuration space**



15,000+ options



TRISTATE	61.63	3 <sup>9000</sup>
BOOL	36.40	
INT	1.54	2 <sup>6000</sup>
STRING	0.29	
HEX	0.14	

Linux 5.2.8, arm  
(% of types' options)

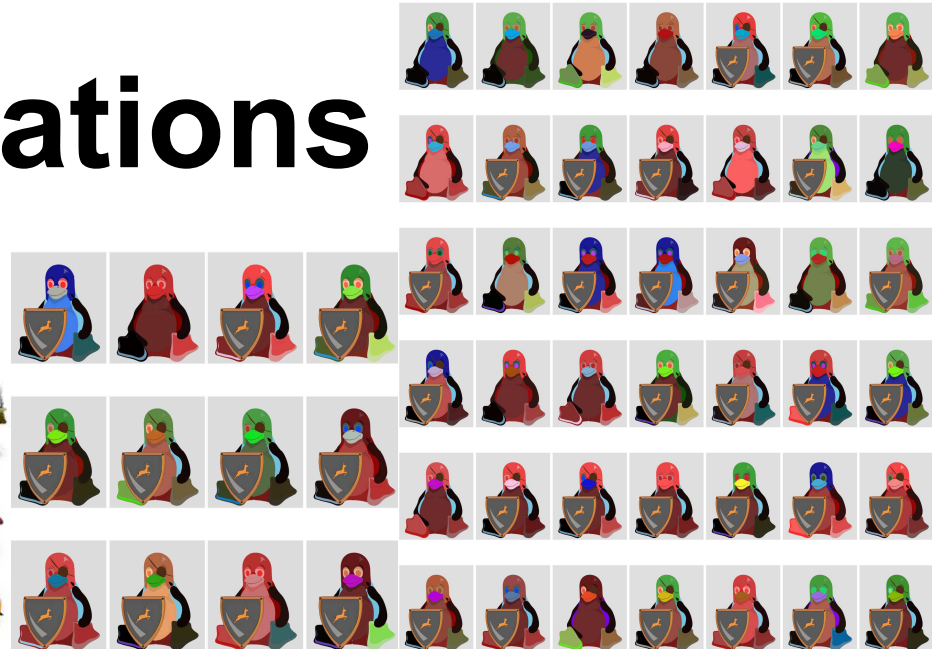
≈ 10<sup>6000</sup> configurations  
(without constraints)



# Linux Kernel

≈ 10<sup>6000</sup>

# configurations

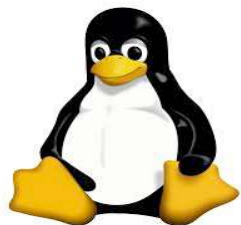


# Linux Kernel

$\approx 10^{6000}$  configurations

$\approx 10^{80}$  is the estimated number of atoms in the universe

$\approx 10^{40}$  is the estimated number of possible chess positions

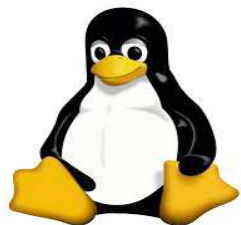


# Linux vs AlphaZero

Building a kernel configuration takes **10 minutes on average** on a recent machine

Trial and error is **cheap** for Chess/Go, you can experience winning/losing billions of time



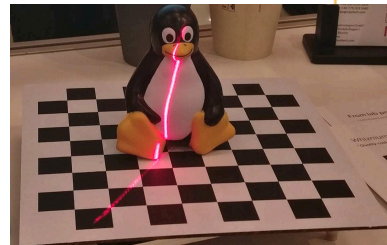


# AlphaZero vs Linux

In Chess/Go, you can fully observe the outcome, without noise and with a **perfect simulator**

Think about **technically measuring the boot time** of a kernel out of a configuration

# Is taming the Linux kernel configuration space a harder problem than resolving Chess?



# configurations

$\approx 10^{6000}$

$\approx 10^{40}$

exploration

costly and  
hard to  
engineer

cheap with a  
perfect  
simulator

# You cannot build $\approx 10^{6000}$ configurations.

## TUXML: predicting out of a (small) sample of configurations' kernels

```
# Processor type and features
#
# CONFIG_ZONE_DMA is not set
# CONFIG_SMP is not set
# CONFIG_X86_FEATURE_NAMES is not set
# CONFIG_X86_FAST_FEATURE_TESTS is not set
CONFIG_X86_MPPARSE=y
CONFIG_X86_X2APIC=y
CONFIG_GOLDFISH=y
# CONFIG_INTEL_RDT_A is not set
# CONFIG_X86_EXTENDED_PLATFORM is not set
CONFIG_IOSF_MBI=m
CONFIG_IOSF_MBI_DEBUG=y
CONFIG_X86_SUPPORTS_MEMORY_FAILURE=y
# CONFIG_SCHED_OMIT_FRAME_POINTER is not set
```

build passing

```
CONFIG_PM_WAKELOCKS=y
CONFIG_PM_WAKELOCKS_LIMIT=100
CONFIG_PM_WAKELOCKS_GC=y
CONFIG_PM=y
# CONFIG_PM_DEBUG is not set
CONFIG_PM_CLK=y
CONFIG_PM_GENERIC_DOMAINS=y
CONFIG_WQ_POWER_EFFICIENT_DEFAULT=y
CONFIG_PM_GENERIC_DOMAINS_SLEEP=y
CONFIG_PM_GENERIC_DOMAINS_OF=y
CONFIG_ENERGY_MODEL=y
CONFIG_ARCH_SUPPORTS_ACPI=y
```

build passing

```
CONFIG_VM_EVENT_COUNTERS=y
CONFIG_SLUB_DEBUG=y
# CONFIG_SLUB_MEMCG_SYSFS_ON is not set
# CONFIG_COMPAT_BRK is not set
# CONFIG_SLAB is not set
CONFIG_SLUB=y
# CONFIG_SLOB is not set
# CONFIG_SLAB_MERGE_DEFAULT is not set
# CONFIG_SLAB_FREELIST_RANDOM is not set
# CONFIG_SLAB_FREELIST_HARDENED is not set
CONFIG_SLAB_FREELIST_HARDENED=y
CONFIG_SLAB_FREELIST_HARDENED_DEBUG=y
```



```
CONFIG_PM_WAKELOCKS=y
CONFIG_PM_WAKELOCKS_LIMIT=100
CONFIG_PM_WAKELOCKS_GC=y
CONFIG_PM=y
# CONFIG_PM_DEBUG is not set
CONFIG_PM_CLK=y
CONFIG_PM_GENERIC_DOMAINS=y
CONFIG_WQ_POWER_EFFICIENT_DEFAULT=y
CONFIG_PM_GENERIC_DOMAINS_SLEEP=y
CONFIG_PM_GENERIC_DOMAINS_OF=y
CONFIG_ENERGY_MODEL=y
CONFIG_ARCH_SUPPORTS_ACPI=y
```

build passing



Classification problem:  
predict the class (BUILD/FAILURE)  
out of options values

# You cannot build $\approx 10^{6000}$ configurations.

## TUXML: predicting out of a (small) sample of configurations' kernels

```
# Processor type and features
#
# CONFIG_ZONE_DMA is not set
# CONFIG_SMP is not set
# CONFIG_X86_FEATURE_NAMES is not set
# CONFIG_X86_FAST_FEATURE_TESTS is not set
CONFIG_X86_X2APIC=y
CONFIG_X86_MPPARSE=y
CONFIG_GOLDFISH=y
# CONFIG_INTEL_RDT_A is not set
# CONFIG_X86_EXTENDED_PLATFORM is not set
CONFIG_IOSF_MBI=m
CONFIG_IOSF_MBI_DEBUG=y
CONFIG_X86_SUPPORTS_MEMORY_FAILURE=y
# CONFIG_SCHED_OMIT_FRAME_POINTER is not set
```

7.1Mb

```
CONFIG_PM_WAKELOCKS=y
CONFIG_PM_WAKELOCKS_LIMIT=100
CONFIG_PM_WAKELOCKS_GC=y
CONFIG_PM=y
# CONFIG_PM_DEBUG is not set
CONFIG_PM_CLK=y
CONFIG_PM_GENERIC_DOMAINS=y
CONFIG_WQ_POWER_EFFICIENT_DEFAULT=y
CONFIG_PM_GENERIC_DOMAINS_SLEEP=y
CONFIG_PM_GENERIC_DOMAINS_OF=y
CONFIG_ENERGY_MODEL=y
CONFIG_ARCH_SUPPORTS_ACPI=y
```

176.8Mb

```
CONFIG_VM_EVENT_COUNTERS=y
CONFIG_SLUB_DEBUG=y
# CONFIG_SLUB_MEMCG_SYSFS_ON is not set
# CONFIG_COMPAT_BRK is not set
# CONFIG_SLAB is not set
CONFIG_SLUB=y
# CONFIG_SLOB is not set
# CONFIG_SLAB_MERGE_DEFAULT is not set
# CONFIG_SLAB_FREELIST_RANDOM is not set
# CONFIG_SLAB_FREELIST_HARDENED is not set
CONFIG_SHUFFLE_PAGE_ALLOCATOR=y
CONFIG_SLUB_CPU_PARTIAL=y
CONFIG_SYSTEM_DATA_VERIFICATION=y
```

16.1Mb

```
CONFIG_PM_WAKELOCKS=y
CONFIG_PM_WAKELOCKS_LIMIT=100
CONFIG_PM_WAKELOCKS_GC=y
CONFIG_PM=y
# CONFIG_PM_DEBUG is not set
CONFIG_PM_CLK=y
CONFIG_PM_GENERIC_DOMAINS=y
CONFIG_WQ_POWER_EFFICIENT_DEFAULT=y
CONFIG_PM_GENERIC_DOMAINS_SLEEP=y
CONFIG_PM_GENERIC_DOMAINS_OF=y
CONFIG_ENERGY_MODEL=y
CONFIG_ARCH_SUPPORTS_ACPI=y
```

102.3Mb



Regression problem:  
predict a quantitative value (eg size)  
out of options values

You cannot build  $\approx 10^{6000}$  configurations.  
**TUXML: predicting out of a (small)  
sample of configurations' kernels**

```
# Processor type and featur  
#  
# CONFIG_ZON  
# CONFIG_SMP  
# CONFIG_X86  
# CONFIG_X86  
CONFIG_X86_X  
CONFIG_X86_M  
CONFIG_GOLDF  
# CONFIG_INT  
# CONFIG_X86  
CONFIG_IOSF_MBI=m  
CONFIG_IOSF_MBI_DEBUG=y  
CONFIG_X86_SUPPORTS_MEMORY_FAILURE=y  
# CONFIG_SCHED_OMIT_FRAME_POINTER is not set
```



**You cannot build  $\approx 10^{6000}$  configurations.  
Is machine learning effective for such  
very large configurable systems?**

```
# Processor type and featur  
#  
# CONFIG_ZON  
# CONFIG_SMP  
# CONFIG_X86  
# CONFIG_X86  
CONFIG_X86_X  
CONFIG_X86_M  
CONFIG_GOLDF  
# CONFIG_INT  
# CONFIG_X86  
CONFIG_IOSF_MBI=m  
CONFIG_IOSF_MBI_DEBUG=y  
CONFIG_X86_SUPPORTS_MEMORY_FAILURE=y  
# CONFIG_SCHED_OMIT_FRAME_POINTER is not set
```



# Answers in the rest of the talk

- Sampling and Learning with TUXML
- Results over 150K+ configurations
  - build failure understanding and prevention
  - kernel size prediction
- Challenges
  - “smart” build infrastructure
  - with devs/contributors in the loop

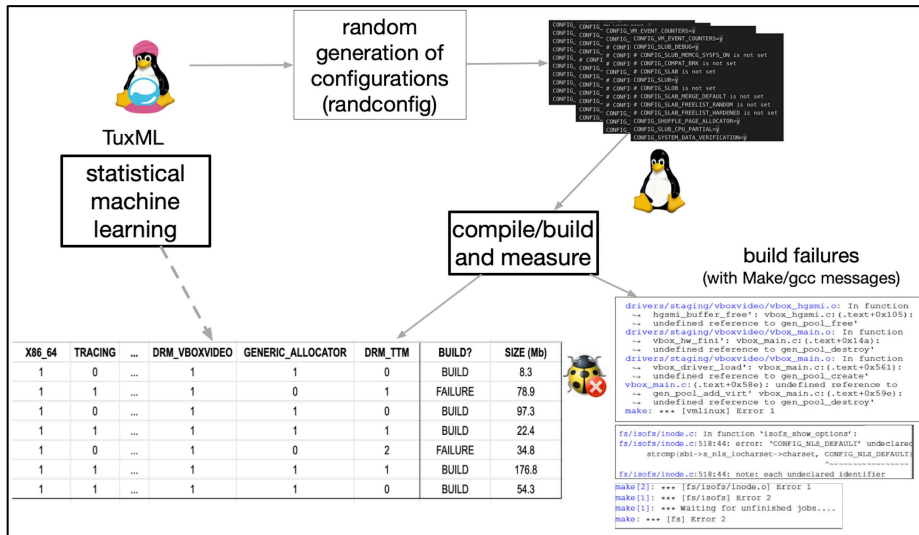






# TUXML: Sampling, Measuring, Learning

<https://github.com/TuxML/>



**Docker** for a reproducible environment with tools/packages needed and **Python** procedures inside

Easy to launch campaign:  
"python kernel\_generator.py 10"

builds/measures  
10 random configurations  
(information sent to a database)

# TUXML: Sampling, Measuring, Learning

<https://github.com/TuxML/>

cid	compilation_date	compilation_time	config_file	stdout_log_file	stderr_log_file	user_output_file	compiled_kernel_size	compressed_compiled_kernel_size
74882	2019-08-12 17:09:42	399.856	[BLOB - 24,3 Kio]	[BLOB - 33,7 Kio]	[BLOB - 14 o]	[BLOB - 702 o]	74559280	GZIP-bzImage : 8855504 , GZIP-vmlinux : 10943304 , ...
74881	2019-08-12 16:58:09	460.392	[BLOB - 25,8 Kio]	[BLOB - 34,7 Kio]	[BLOB - 14 o]	[BLOB - 704 o]	81377768	GZIP-bzImage : 18375632 , GZIP-vmlinux : 20462408 ...
74880	2019-08-12 16:47:28	301.775	[BLOB - 22 Kio]	[BLOB - 24,2 Kio]	[BLOB - 14 o]	[BLOB - 705 o]	83004496	GZIP-bzImage : 14365648 , GZIP-vmlinux : 16452424 ...
74879	2019-08-12 16:46:14	1393.61	[BLOB - 24,1 Kio]	[BLOB - 50 Kio]	[BLOB - 571 o]	[BLOB - 712 o]	109098328	GZIP-bzImage : 17183792 , GZIP-vmlinux : 19272160 ...
74878	2019-08-12 16:45:03	305.705	[BLOB - 26,1 Kio]	[BLOB - 28,8 Kio]	[BLOB - 14 o]	[BLOB - 703 o]	55523752	GZIP-bzImage : 14767568 , GZIP-vmlinux : 16852088 ...

Console de requêtes SQL

(information sent to a database)

# Data: version 4.13.3 and 4.15 (x86\_64)

cid	compilation_date	compilation_time	config_file	stdout_log_file	stderr_log_file	user_output_file	compiled_kernel_size	compressed_compiled_kernel_size
74882	2019-08-12 17:09:42	399.856	[BLOB - 24,3 Kio]	[BLOB - 33,7 Kio]	[BLOB - 14 o]	[BLOB - 702 o]	74559280	GZIP-bzImage : 8855504 , GZIP-vmlinux : 10943304 ...
74881	2019-08-12 16:58:09	460.392	[BLOB - 25,8 Kio]	[BLOB - 34,7 Kio]	[BLOB - 14 o]	[BLOB - 704 o]	81377768	GZIP-bzImage : 18375632 , GZIP-vmlinux : 20462408 ...
74880	2019-08-12 16:47:28	301.775	[BLOB - 22 Kio]	[BLOB - 24,2 Kio]	[BLOB - 14 o]	[BLOB - 705 o]	83004496	GZIP-bzImage : 14365648 , GZIP-vmlinux : 16452424 ...
74879	2019-08-12 16:46:14	1393.61	[BLOB - 24,1 Kio]	[BLOB - 50 Kio]	[BLOB - 571 o]	[BLOB - 712 o]	109098328	GZIP-bzImage : 17183792 , GZIP-vmlinux : 19272160 ...
74878	2019-08-12 16:45:03	305.705	[BLOB - 26,1 Kio]	[BLOB - 28,8 Kio]	[BLOB - 14 o]	[BLOB - 703 o]	55523752	GZIP-bzImage : 14767568 , GZIP-vmlinux : 16852088 ...

74K+ configurations for Linux 4.15

95K+ configurations for Linux 4.13.3

(and 15K hours of computation on a grid computing)

# Application 1: “Smart” build infrastructure



results for 4.13.3 only

cid	compilation_date	compilation_time	config_file	stdout_log_file	stderr_log_file	user_output_file	compiled_kernel_size	compressed_compiled_kernel_size
74882	2019-08-12 17:09:42	399.856	[BLOB - 24.3 Kio]	[BLOB - 33.7 Kio]	[BLOB - 14 o]	[BLOB - 702 o]	74559280	GZIP-bzimage : 8855504, GZIP-vmlinux : 10943304 ...
74881	2019-08-12 16:58:09	460.392	[BLOB - 25.8 Kio]	[BLOB - 34.7 Kio]	[BLOB - 14 o]	[BLOB - 704 o]	81377768	GZIP-bzimage : 18375632, GZIP-vmlinux : 20462408 ...
74880	2019-08-12 16:47:28	301.775	[BLOB - 22 Kio]	[BLOB - 24.2 Kio]	[BLOB - 14 o]	[BLOB - 705 o]	83004496	GZIP-bzimage : 14365648, GZIP-vmlinux : 19462624 ...
74879	2019-08-12 16:46:14	1393.61	[BLOB - 24.1 Kio]	[BLOB - 50 Kio]	[BLOB - 571 o]	[BLOB - 712 o]	109098328	GZIP-bzimage : 17183792, GZIP-vmlinux : 19272160 ...
74878	2019-08-12 16:45:03	305.705	[BLOB - 26.1 Kio]	[BLOB - 28.8 Kio]	[BLOB - 14 o]	[BLOB - 703 o]	55523752	GZIP-bzimage : 14767568, GZIP-vmlinux : 14809888 ...

95,854 configurations  
3,164 configuration failures  
5.83% of build lead to failures

```
drivers/staging/vboxvideo/vbox_hgsmi.o: In function
↳ hgsmi_buffer_free': vbox_hgsmi.c:(.text+0x105):
↳ undefined reference to gen_pool_free'
drivers/staging/vboxvideo/vbox_main.o: In function
↳ vbox_hw_fini': vbox_main.c:(.text+0x14a):
↳ undefined reference to gen_pool_destroy'
drivers/staging/vboxvideo/vbox_main.o: In function
↳ vbox_driver_load': vbox_main.c:(.text+0x561):
↳ undefined reference to gen_pool_create'
vbox_main.c:(.text+0x58e): undefined reference to
↳ gen_pool_add_virt' vbox_main.c:(.text+0x59e):
↳ undefined reference to gen_pool_destroy'
make: *** [vmlinux] Error 1
```

```
fs/isofs/inode.c: In function 'isofs_show_options':
fs/isofs/inode.c:518:44: error: 'CONFIG_NLS_DEFAULT' undeclared
      strcmp(sbi->s_nls_iocharset->charset, CONFIG_NLS_DEFAULT
      ^~~~~~
fs/isofs/inode.c:518:44: note: each undeclared identifier
```

Should we send 3,164 bug reports?

# Application 1: “Smart” build infrastructure

One configuration **bug**  
can lead to  
many configuration  
**failures**

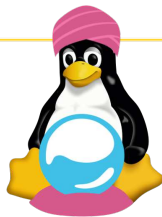
DRM\_VBOXVIDEO &  
GENERIC\_ALLOCATOR



**367** failures like this

```
drivers/staging/vboxvideo/vbox_hgsmi.o: In function
↳ hgsmi_buffer_free': vbox_hgsmi.c:(.text+0x105):
↳ undefined reference to gen_pool_free'
d
drivers/staging/vboxvideo/vbox_hgsmi.o: In function
↳ hgsmi_buffer_free': vbox_hgsmi.c:(.text+0x105):
↳ undefined reference to gen_pool_free'
drivers/staging/vboxvideo/vbox_main.o: In function
↳ vbox_hw_fini': vbox_main.c:(.text+0x14a):
↳ undefined reference to gen_pool_destroy'
di
drivers/staging/vboxvideo/vbox_hgsmi.o: In function
↳ hgsmi_buffer_free': vbox_hgsmi.c:(.text+0x105):
↳ undefined reference to gen_pool_free'
vbox_main.o: In function
↳ vbox_hw_fini': vbox_main.c:(.text+0x14a):
↳ undefined reference to gen_pool_destroy'
ma
drivers/staging/vboxvideo/vbox_hgsmi.o: In function
↳ hgsmi_buffer_free': vbox_hgsmi.c:(.text+0x105):
↳ undefined reference to gen_pool_create'
vbox_main.c:(.text+0x58e): undefined reference to
↳ gen_pool_add_virt' vbox_main.c:(.text+0x59e):
↳ undefined reference to gen_pool_destroy'
make: *** [vmlinux] Error 1
```

Statistical learning can **automatically** pinpoint  
what combinations of options lead to a failure



# Classification problem: predict the class (BUILD/FAILURE) out of options values


```
# Processor type and features
#
# CONFIG_ZONE_DMA is not set
# CONFIG_SMP is not set
# CONFIG_X86_FEATURE_NAMES is not set
# CONFIG_X86_FAST_FEATURE_TESTS is not set
CONFIG_X86_X2APIC=y
CONFIG_X86_MPPARSE=y
CONFIG_GOLDFISH=y
# CONFIG_INTEL_RDT_A is not set
```

build passing

```
CONFIG_PM_WAKELOCKS=y
CONFIG_PM_WAKELOCKS_LIMIT=100
CONFIG_PM_WAKELOCKS_GC=y
CONFIG_PM=y
# CONFIG_PM_DEBUG is not set
CONFIG_PM_CLK=y
CONFIG_PM_GENERIC_DOMAINS=y
CONFIG_WQ_POWER_EFFICIENT_DEFAULT=y
CONFIG_PM_GENERIC_DOMAINS_SLEEP=y
```

build passing

```
CONFIG_VM_EVENT_COUNTERS=y
CONFIG_SLUB_DEBUG=y
# CONFIG_SLUB_MEMCG_SYSFS_ON is not set
# CONFIG_COMPAT_BRK is not set
# CONFIG_SLAB is not set
CONFIG_SLUB=y
# CONFIG_SLOB is not set
# CONFIG_DEFAULT is not set
# CONFIG_LIST_RANDOM is not set
# CONFIG_LIST_HARDENED is not set
# CONFIG_GENERIC_ALLOCATOR=y
# CONFIG_PARTIAL=y
# CONFIG_A_VERIFICATION=y
```



```
CONFIG_PM_WAKELOCKS=y
CONFIG_PM_WAKELOCKS_LIMIT=100
CONFIG_PM_WAKELOCKS_GC=y
CONFIG_PM=y
# CONFIG_PM_DEBUG is not set
CONFIG_PM_CLK=y
CONFIG_PM_GENERIC_DOMAINS=y
CONFIG_WQ_POWER_EFFICIENT_DEFAULT=y
CONFIG_PM_GENERIC_DOMAINS_SLEEP=y
```

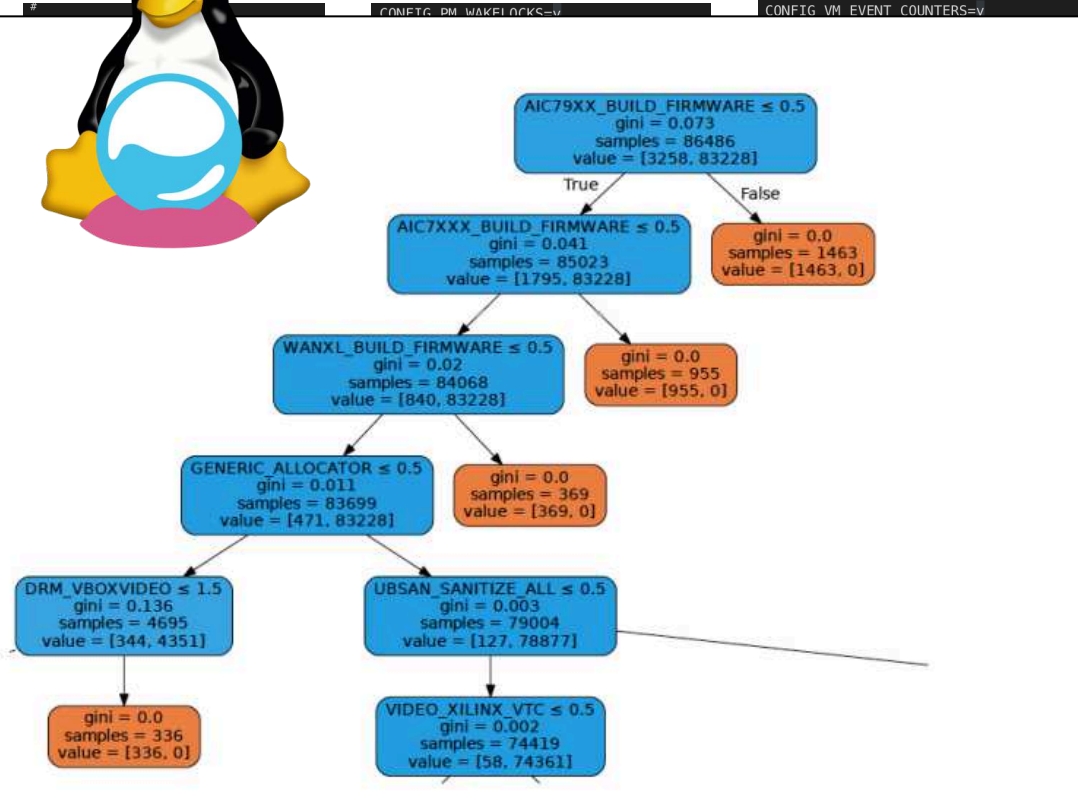
build passing

X86_64	TRACING	...	DRM_VBOXVIDEO	GENERIC_ALLOCATOR	DRM_TTM	BUILD?	SIZE (Mb)
1	0	...	1	1	0	BUILD	8.3
1	1	...	1	0	1	FAILURE	78.9
1	0	...	1	1	0	BUILD	97.3
1	1	...	1	1	1	BUILD	22.4
1	0	...	1	0	2	FAILURE	34.8
1	1	...	1	1	1	BUILD	176.8
1	1	...	1	1	0	BUILD	54.3

Do you recognize a pattern?  
(matrix is 95K rows and 12K columns)



# Classification tree



```
CONFIG_PM_WAKELOCKS=y
CONFIG_PM_WAKELOCKS_LIMIT=100
CONFIG_PM_WAKELOCKS_GC=y
CONFIG_PM=y
# CONFIG_PM_DEBUG is not set
CONFIG_PM_CLK=y
CONFIG_PM_GENERIC_DOMAINS=y
CONFIG_WQ_POWER_EFFICIENT_DEFAULT=y
CONFIG_PM_GENERIC_DOMAINS_SLEEP=y
y
build passing
```

	BUILD?	SIZE (Mb)
	BUILD	8.3
	FAILURE	78.9
	BUILD	97.3
	BUILD	22.4
	FAILURE	34.8
	BUILD	176.8
	BUILD	54.3

tern?  
nns)

# Application 1: “Smart” build infrastructure

```
X86_64=y
USB=m
...
# always leads to a failure
AIC7XXX_BUILD_FIRMWARE=y
...
# always leads to a failure
DRM_VBOXVIDEO=y
GENERIC_ALLOCATOR=n
```

```
aicasm_symbol.c:48:19: fatal error: aicdb.h: No such
  ↪ file or directory
#include "aicdb.h"
make[4]: *** [aicasm] Error 1
make[3]: *** [drivers/scsi/aic7xxx/aicasm/aicasm]
  ↪ Error 2
make[2]: *** [drivers/scsi/aic7xxx] Error 2
make: *** [drivers] Error 2
```

(a) Configuration failure due to AIC7XXX\_BUILD\_FIRMWARE

```
X86_64=y
USB=y
...
AIC7XXX_BUILD_FIRMWARE=n
...
# always leads to a failure
DRM_VBOXVIDEO=y
GENERIC_ALLOCATOR=n
```

```
drivers/staging/vboxvideo/vbox_hgsmi.c: In function
  ↪ hgsmi_buf_for_free': vbox_hgsmi.c:(.text+0x105):
  ↪ undefined reference to gen_pool_free'
drivers/staging/vboxvideo/vbox_main.c: In function
  ↪ vbox_hw_fini': vbox_main.c:(.text+0x14a):
  ↪ undefined reference to gen_pool_destroy'
drivers/staging/vboxvideo/vbox_main.o: In function
  ↪ vbox_driver_load': vbox_main.c:(.text+0x561):
  ↪ undefined reference to gen_pool_create'
vbox_main.o:(.text+0x58e): Undefined reference to
  ↪ gen_pool_add_virt' vbox_main.c:(.text+0x59e):
  ↪ undefined reference to gen_pool_destroy'
make: *** [vmlinux] Error 1
```

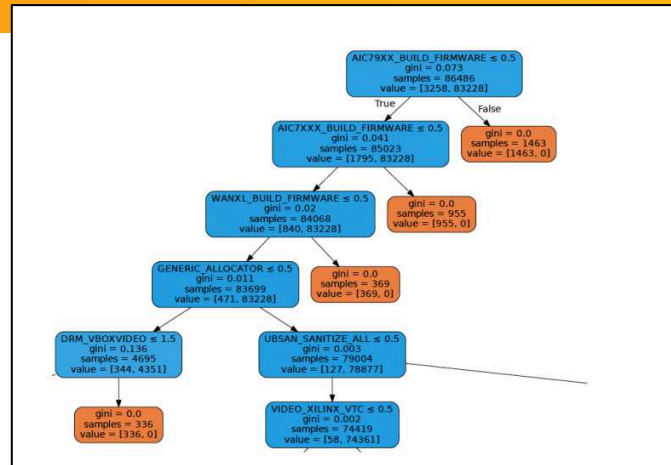
(b) configuration failure due to DRM\_VBOXVIDEO, GENERIC\_ALLOCATOR

```
X86_64=y
# always leads to a failure
AIC7XXX_BUILD_FIRMWARE=y
# always leads to a failure
FORTIFY_SOURCE=y
UBSAN_NULL=y
UBSAN_SANITIZE_ALL=y
IPV6=y
INFINIBAND_ADDR_TRANS=y
...
```

```
./include/linux/string.h:305:4: error: call to
  ↪ __read_overflow2' declared with attribute
  ↪ error: detected read beyond size of object
  ↪ passed as 2nd parameter
  ↪ __read_overflow2();
make[3]: ***
  ↪ [drivers/infiniband/core/roce_gid_mgmt.o] Error
  ↪ 1
make[2]: *** [drivers/infiniband/core] Error 2
make[1]: *** [drivers/infiniband] Error 2
make: *** [drivers] Error 2
```

(c) Configuration failure due to FORTIFY\_SOURCE +options in red

Figure 2: Configuration failures and error messages. Which options' values cause the failures? How to prevent failures?



## Some config. bugs can mask/ dominate other config. bugs!

Solution (see paper): statistical learning combined with clustering of error messages (multi-class classification)



# Application 1: “Smart” build infrastructure

**5.83%** of build failures can be explained by **16** config. bugs of Linux and **3** config. bugs of TUXML

nb_failures	percentage	bug (faulty option)	Bug?	Fix
2464	68.05	AIC7XXX_BUILD_FIRMWARE   AIC79XX_BUILD_FIRMWARE	TUXML	missing tools / Kconfig doc.
476	13.15	WANXL_BUILD_FIRMWARE	TUXML	missing tools / Kconfig doc.
367	10.14	DRM_VBOXVIDEO & GENERIC_ALLOCATOR	Linux	Kconfig dependency
161	4.45	AIC7XXX_BUILD_FIRMWARE   AIC79XX_BUILD_FIRMWARE	TUXML	missing tools / Kconfig doc.
83	2.29	FORTIFY_SOURCE & UBSAN_SANITIZE_ALL & INFINIBA...	Linux	source code
19	0.52	VIDEO_MUX & VIDEO_V4L2	Linux	Kconfig dependency
15	0.41	BACKLIGHT_CLASS_DEVICE & DRM_I915   DRM_SAVAGE...	Linux	Kconfig dependency
13	0.36	DRM_VBOXVIDEO & DRM_TTM	Linux	Kconfig dependency
6	0.17	NLS & ..	Linux	source code
3	0.08	SPI_JCORE & ..	Linux	source code
3	0.08	GPIOLIB & ..	Linux	Kconfig dependency
2	0.06	CRC32 & VIDEO	Linux	Kconfig dependency
2	0.06	BT_HCIUART_H4 & ..	Linux	Kconfig dependency
2	0.06	REGMAP_MMIO & ..	Linux	Kconfig dependency
1	0.03	VIDEO_SAA7134_GO7007 & SND_SOC_RT5514_SPI	Linux	Kconfig dependency
1	0.03	USB_F_TCM & ..	Linux	Kconfig dependency
1	0.03	VIDEO_SOLO6X10 & ..	Linux	source code
1	0.03	VIDEO_ATOMISP & ...	Linux	source code + Kconfig dep.
1	0.03	NEW_LEDS & ..	Linux	Kconfig dependency

# Application 1: “Smart” build infrastructure

**5.83%** of build failures can be explained by  
**16** config. bugs of Linux and **3** config. bugs of TUXML

Don't trust your configuration build infrastructure!  
Prevent/Fix as early as possible configuration bugs  
(otherwise you won't see other bugs!)

Bug **location/understanding**: TUXML can help to pinpoint  
responsible options (and avoid sending duplicate bugs)

TUXML can **prevent failures** and avoid building buggy  
configs (until a fix is done) with a good accuracy

# Application 2: Kernel Size Prediction

[https://elinux.org/Kernel\\_Size\\_Tuning\\_Guide](https://elinux.org/Kernel_Size_Tuning_Guide) - Tim Bird (Sony)

Linux kernel tinification/tinyconfig - Josh Triplett (Intel)

Challenges of Low Spec Embedded Linux - Alexander Sack, Pantacor

Timing Boot Time Reduction Techniques - Michael Opdenacker, Bootlin  
@ ELC 2019

**Unfortunately, nobody knows the precise effect of (combinations of) options on size**

Kconfig: (only?) 150 options are explicitly referring to size





# Regression problem: predict a quantitative value (eg size) out of options values

```
# Processor type and features
#
# CONFIG_ZONE_DMA is not set
# CONFIG_SMP is not set
# CONFIG_X86_FEATURE_NAMES is not set
# CONFIG_X86_FAST_FEATURE_TESTS is not set
CONFIG_X86_X2APIC=y
CONFIG_X86_MPPARSE=y
CONFIG_GOLDFISH=y
# CONFIG_INTEL_RDT_A is not set
# CONFIG_X86_EXTENDED_PLATFORM is not set
CONFIG_IO5F_MBI=m
CONFIG_IO5F_MBI_DEBUG=y
CONFIG_X86_SUPPORTS_MEMORY_FAILURE=y
# CONFIG_SCHED_OMIT_FRAME_POINTER is not set
```

7.1Mb

```
CONFIG_PM_WAKELOCKS=y
CONFIG_PM_WAKELOCKS_LIMIT=100
CONFIG_PM_WAKELOCKS_GC=y
CONFIG_PM=y
# CONFIG_PM_DEBUG is not set
CONFIG_PM_CLK=y
CONFIG_PM_GENERIC_DOMAINS=y
CONFIG_WQ_POWER_EFFICIENT_DEFAULT=y
CONFIG_PM_GENERIC_DOMAINS_SLEEP=y
CONFIG_PM_GENERIC_DOMAINS_OF=y
CONFIG_ENERGY_MODEL=y
CONFIG_ARCH_SUPPORTS_ACPI=y
```

176.8Mb

```
CONFIG_VM_EVENT_COUNTERS=y
CONFIG_SLUB_DEBUG=y
# CONFIG_SLUB_MEMCG_SYSFS_ON is not set
# CONFIG_COMPAT_BRK is not set
# CONFIG_SLAB is not set
CONFIG_SLUB=y
# CONFIG_SLOB is not set
# CONFIG_SLAB_MERGE_DEFAULT is not set
# CONFIG_SLAB_FREELIST_RANDOM is not set
# CONFIG_SLAB_FREELIST_HARDENED is not set
CONFIG_SHUFFLE_PAGE_ALLOCATOR=y
CONFIG_SLUB_CPU_PARTIAL=y
CONFIG_SYSTEM_DATA_VERIFICATION=y
```

16.1Mb

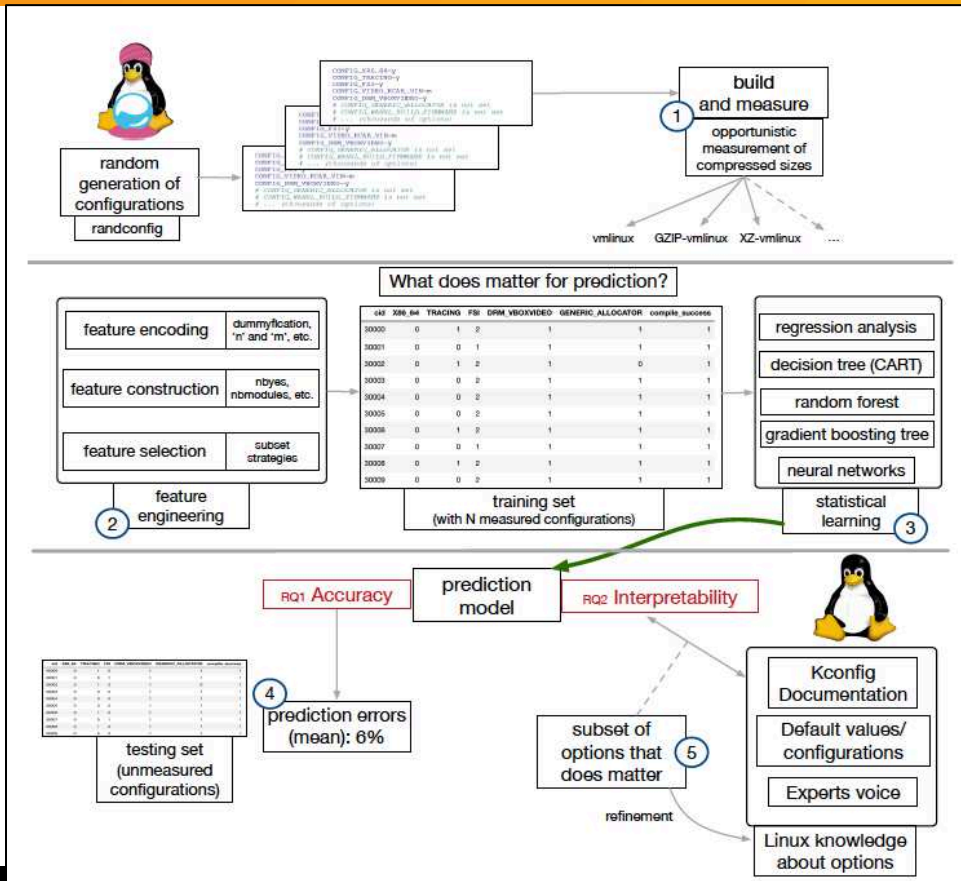
```
CONFIG_PM_WAKELOCKS=y
CONFIG_PM_WAKELOCKS_LIMIT=100
CONFIG_PM_WAKELOCKS_GC=y
CONFIG_PM=y
# CONFIG_PM_DEBUG is not set
CONFIG_PM_CLK=y
CONFIG_PM_GENERIC_DOMAINS=y
CONFIG_WQ_POWER_EFFICIENT_DEFAULT=y
CONFIG_PM_GENERIC_DOMAINS_SLEEP=y
CONFIG_PM_GENERIC_DOMAINS_OF=y
CONFIG_ENERGY_MODEL=y
CONFIG_ARCH_SUPPORTS_ACPI=y
```

102.3Mb

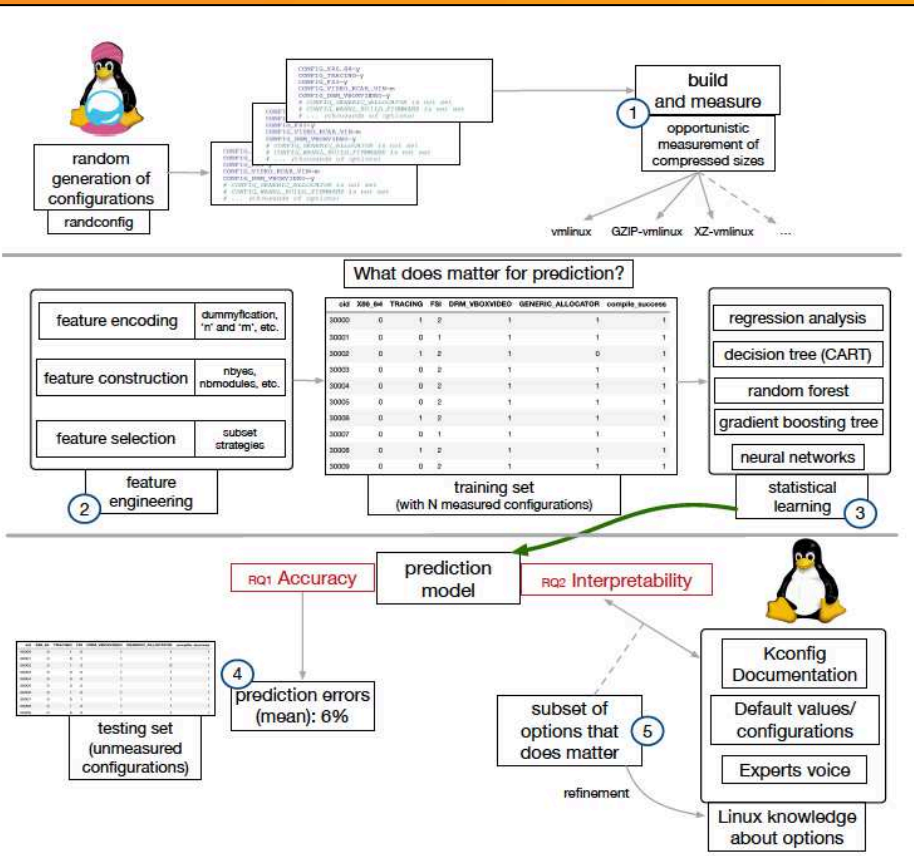
**Smart configuration:** prediction model can quantify the effect of (de-)activating options (optimizer/recommender/configurator can be built on top of it)

Documentation/default config. improvement: identification of **“influential”** options

# Application 2: Kernel Size Prediction



# Application 2: Kernel Size Prediction



## vmlinux and compressed sizes

	GZIP	BZIP2	LZMA	XZ	LZO	LZ4
<b>GZiPo</b>	0	-28.1465	17.6087	25.5951	-7.48979	-12.3437
<b>BZiPo</b>	41.0556	0	65.3317	76.2405	30.6506	23.8874
<b>LZMAo</b>	-14.8725	-39.0396	0	6.74528	-21.2054	-25.3171
<b>XZo</b>	-20.0819	-42.8681	-6.15874	0	-26.0152	-29.8666
<b>LZOo</b>	8.14584	-22.1959	27.2664	35.9318	0	-5.26847
<b>LZ4o</b>	14.2047	-17.7728	34.4445	43.6185	5.57607	0

config KERNEL\_XZ

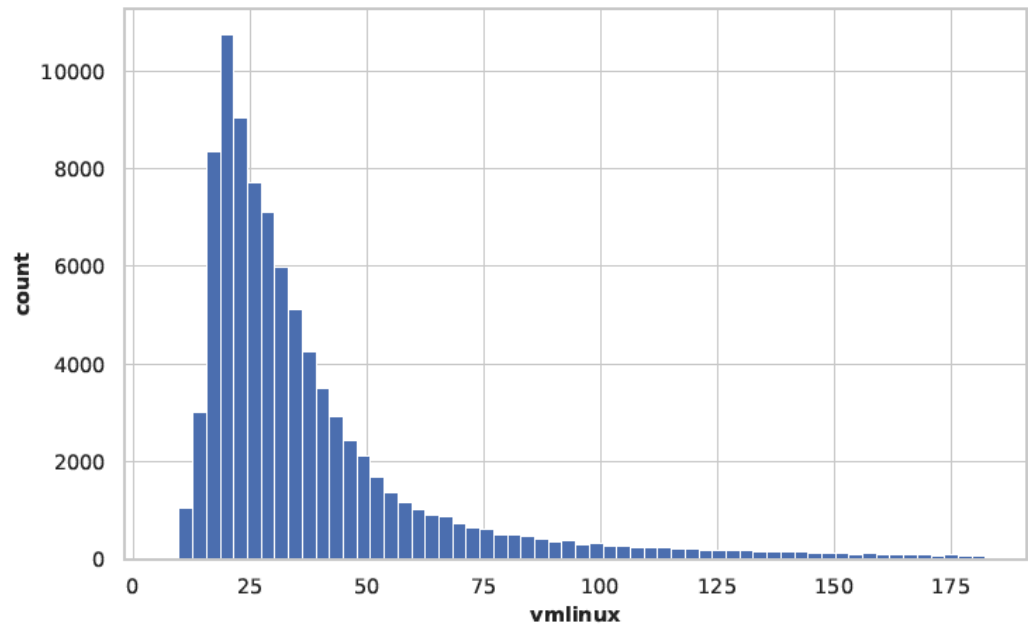
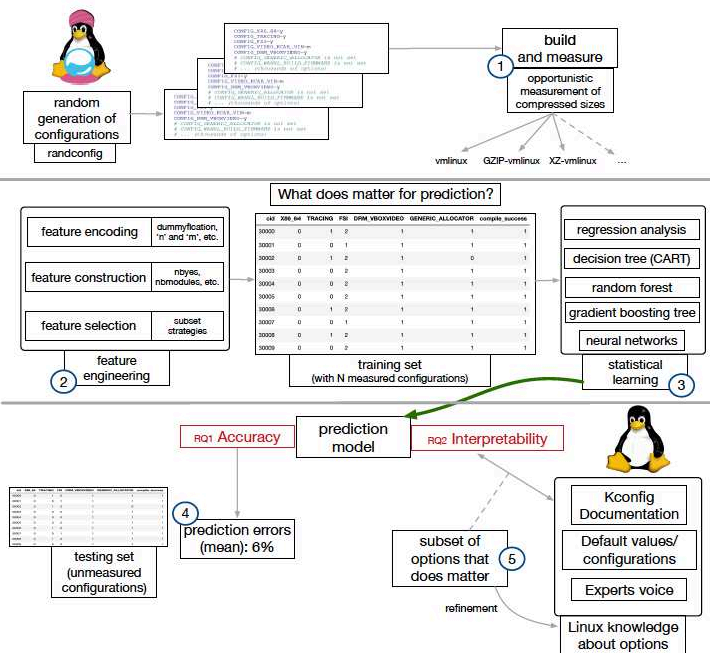
bool "XZ"

depends on HAVE\_KERNEL\_XZ

help

XZ uses the LZMA2 algorithm and instruction set specific BJC filters which can improve compression ratio of executable code. The size of the kernel is about 30% smaller with XZ in comparison to gzip. On architectures for which there is a BJC filter (i386, x86\_64, ARM, IA-64, PowerPC, and SPARC), XZ will create a few percent smaller kernel than plain LZMA.

# Application 2: Kernel Size Prediction



(size of vmlinux)  
Max: 1,698.14Mb  
Min: 7Mb (tinyconfig)

Figure 4: Distribution of size (in Mb) without outliers



# Application 2: Kernel Size Prediction

Algorithm	Without Feature Selection					With Feature Selection				
	N=10	N=20	N=50	N=80	N=90	N=10	N=20	N=50	N=80	N=90
OLS Regression	74.54±2.3	68.76±1.03	61.9±1.14	50.37±0.57	49.42±0.08	43.56±1.48	42.58±2.22	40.23±0.22	39.56±0.39	39.29±0.48
Lasso	34.13±1.38	34.32±0.12	36.58±1.04	38.07±0.08	38.04±0.17	35.18±0.45	36.53±0.6	39.28±1.06	38.28±0.04	38.61±0.81
Ridge	139.63±1.13	91.43±1.07	62.42±0.08	55.75±0.2	51.78±0.14	43.52±1.41	42.29±2.16	40.2±0.27	39.53±0.33	39.24±0.43
ElasticNet	79.26±0.9	80.81±1.05	80.58±0.77	80.57±0.71	80.34±0.53	79.66±2.11	81.74±0.65	81.0±0.24	80.84±0.6	81.45±0.2
Decision Tree	15.18±0.13	13.21±0.12	11.32±0.07	10.61±0.10	10.48±0.15	13.97±0.08	12.34±0.08	10.75±0.05	10.07±0.09	9.91±0.12
Random Forest	12.5±0.19	10.75±0.07	9.27±0.07	8.6±0.07	8.4±0.07	10.79±0.15	9.6±0.08	8.4±0.05	7.96±0.06	7.8±0.05
GB Tree	11.13±0.23	9.43±0.07	7.70±0.04	7.02±0.05	6.83±0.10	8.67±0.09	7.60±0.08	6.65±0.03	6.33±0.03	6.24±0.06
N. Networks	16.73 ±1.30	11.38 ±0.27	9.34 ±0.17	8.11 ±0.26	7.76 ±0.10	14.20 ±0.02	8.7 ±0.06	6.61 ±0.02	5.73 ±0.03	5.52 ±0.12
Polynomial Reg.	-	-	-	-	-	24,65±1.23	22.58±0.18	20.49±0.24	21.53±0.1	20.86±0.04

Table 1: MAPE of different learning algorithms for the prediction of vmlinux size, without and with feature selection

We find a sweet spot where only 200–300 features are sufficient to efficiently train a random forest and a Gradient Boosting Tree to obtain a prediction model that outperforms other baselines (7% prediction errors for 40K configurations). We observe similar feature selection benefits for any training set size and tree-based learning algorithms.

up to ~3% for  
compressed  
kernels sizes!

**Towards smart  
configuration assistant  
(optimizer/recommender/  
configurator)**



# Session 2: Kernel Size Prediction

Thanks to our prediction model, we have effectively identified a list of important features that is consistent with the options and strategy of tinyconfig, the Kconfig documentation, and Linux knowledge. We also found options that can be used to refine or augment the documentation.

**Towards improved documentation/default config. and informed configurations' decisions**

DEBUG\_INFO  
#yes  
DEBUG\_INFO\_REDUCED  
DEBUG\_INFO\_SPLIT  
X86\_NEED\_RELOCS  
RANDOMIZE\_BASE  
UBSAN\_SANITIZE\_ALL  
KASAN  
UBSAN\_ALIGNMENT  
GCOV\_PROFILE\_ALL  
XFS\_DEBUG  
DRM\_NOUVEAU  
XFS\_FS  
KCOV\_INSTRUMENT\_ALL  
DRM\_RADEON  
UBSAN\_NULL  
MAXSMP  
BLK\_MQ\_PCI  
DRM\_AMDGPU  
SCSI\_ISCSI\_ATTRS  
MDIO  
X86\_VSMP

Jild  
teasun  
turistic  
ement o  
ssed size  
Z-vmlinux

reg  
deci  
grad  
fn

Dc  
Default values/  
configurations  
Experts voice  
Linux knowledge  
about options

# Challenges

- Retrospectively and despite our investment, we found relatively few bugs of Linux
  - Is it due to the way we sample?
  - Is it due to the stable version of Linux we chose?
  - Is it due to the high-quality of Linux, its contributors and its industry-strength, community-based effort?

# Challenges

- Sampling is based on randconfig
  - randconfig does not produce uniform random samples
  - hypothesis: the testing “community” has over-fitted randconfig
- We need other sampling strategies!
  - Uniform (but SAT-based techniques should be improved)
  - Coverage-based sampling (e.g., t-wise)
  - Knowledge-based sampling

# Challenges

- The cost of gathering data is important (15K+ hours of computation)
- Incremental build of configurations
- Bugs do not transfer well
- However, kernel size “knowledge” may transfer
  - Instead of starting from scratch, we can transfer a prediction model for another version of Linux (ongoing work)

# Challenges

- Kernel CI / 0-day
  - Our focus: testing **configurations** in the large
  - Complementary!
  - **Learning** techniques can be used in both contexts
  - Sharing data
- Unify the force!

# Challenges

- “Smart” build infrastructure
  - Other properties (e.g., warnings, boot, security)
- With devs/contributors in the loop
  - We need knowledge to validate our learning model
  - We need knowledge to apply “smart” sampling
  - We aim to produce actionable knowledge
- TUXML needs you!



# Conclusion (feedbacks welcome!)

- Learning From Thousands of Build Failures of Linux Kernel Configurations

- Mathieu Acher, Hugo Martin, Juliana Alves Pereira, Arnaud Blouin, Djamel Eddine Khelladi, Jean-Marc Jézéquel
- <https://hal.inria.fr/hal-02147012>

- Learning Very Large Configuration Spaces: What Matters for Linux Kernel Sizes

- Mathieu Acher, Hugo Martin, Juliana Pereira, Arnaud Blouin, Jean-Marc Jézéquel, Djamel Eddine Khelladi, Luc Lesoil, Olivier Barais
- <https://hal.inria.fr/hal-02314830>



# Some related work

- Julia Lawall and Gilles Muller “JMake: Dependable Compilation for Kernel Janitors.” In 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2017
- Iago Abal, Claus Brabrand, and Andrzej Wasowski “42 variability bugs in the linux kernel: a qualitative analysis”. In ACM/IEEE International Conference on Automated Software Engineering, ASE’14
- Jean Melo, Elvis Flesborg, Claus Brabrand, and Andrzej Wasowski “A Quantitative Analysis of Variability Warnings in Linux”. In Proceedings of the Tenth International Workshop on Variability Modelling of Software-intensive Systems (VaMoS’16)
- Sarah Nadi, Thorsten Berger, Christian Kästner, and Krzysztof Czarnecki “Where Do Configuration Constraints Stem From? An Extraction Approach and an Empirical Study” IEEE Trans. Software Eng., 2016
- Minghui Zhou, Qingying Chen, Audris Mockus, and Fengguang Wu “On the Scalability of Linux Kernel Maintainers’ Work”. In Proceedings of the 2017 11<sup>th</sup> Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2017)

# Some related work

- Axel Halin, Alexandre Nuttinck, Mathieu Acher, Xavier Devroey, Gilles Perrouin, Benoit Baudry: Test them all, is it worth it? Assessing configuration sampling on the JHipster Web development stack. *Empirical Software Engineering* 24(2): 674-717 (2019)
- Quentin Plazar, Mathieu Acher, Gilles Perrouin, Xavier Devroey, Maxime Cordy: Uniform Sampling of SAT Solutions for Configurable Systems: Are We There Yet? *ICST 2019*: 240-251
- Juliana Alves Pereira, Hugo Martin, Mathieu Acher, Jean-Marc Jézéquel, Goetz Botterweck, Anthony Ventresque: Learning Software Configuration Spaces: A Systematic Literature Review. *CoRR abs/1906.03018* (2019)
- Paul Temple, Mathieu Acher, Jean-Marc Jézéquel, Olivier Barais: Learning Contextual-Variability Models. *IEEE Software* 34(6): 64-70 (2017)
- Austin Mordahl, Jeho Oh, Ugur Koc, Shiyi Wei, Paul Gazzillo: An empirical study of real-world variability bugs detected by variability-oblivious tools. *ESEC/SIGSOFT FSE 2019*: 50-61

# Thanks!



- DiverSE research team <http://diverse-team.fr>
  - Hugo Martin, Juliana Alves Pereira, Arnaud Blouin, Jean-Marc Jézéquel, Djamel Eddine Khelladi, Luc Lesoil, Olivier Barais
- TUXML team at ISTIC / University of Rennes 1
  - Paul Saffray, Alexis Le Masle, Michaël Picard, Corentin Chédotal, Gwendal Didot, Dorian Dumanget, Antonin Garret, Erwan Le Flem, Pierre Le Luron, Mickaël Lebreton, Fahim Merzouk, Valentin Petit, Julien Royon Chalendard, Cyril Hamon, Luis Thomas, Alexis Bonnet
- IGRIDA <http://igrida.gforge.inria.fr>
- Tim Bird (Sony) and Greg Kroah-Hartman (Linux foundation)
- Julia Lawall (for challenging us to attend ELC!)

# Intrigued by Tux logos?

## Have a look and don't hesitate to contribute!

<https://github.com/diverse-project/tuxart>

Side project: Tux generator out of arbitrary Linux kernel configurations (.config)



Khaled Arsalane

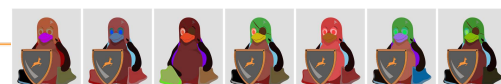
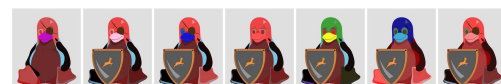
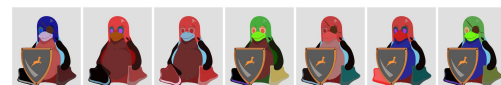
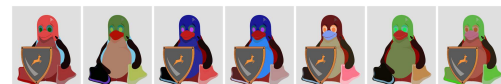
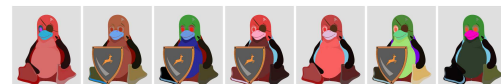
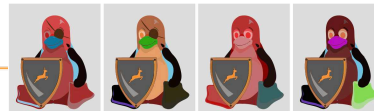
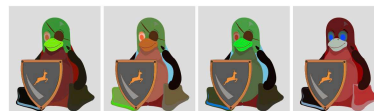
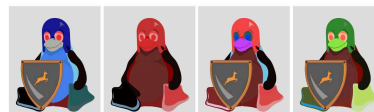
Eliot Marie

Pierre Pouteau

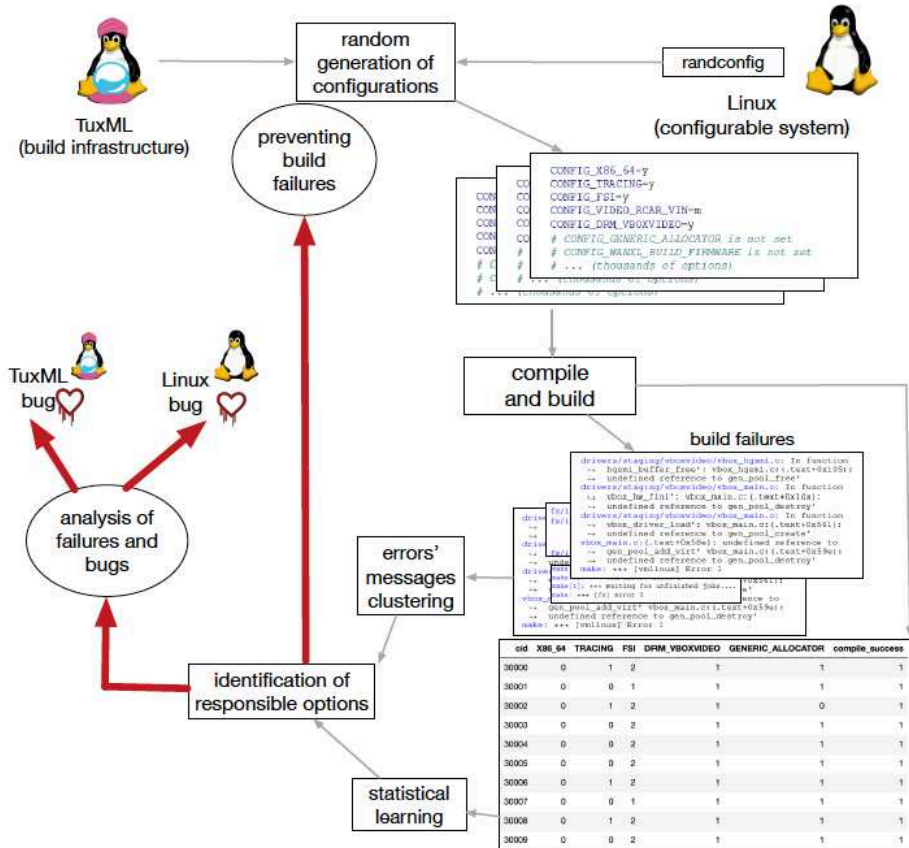
Zakariae Boukhchen

Richard Faraji-Huon

Mathieu Acher



# Application 1: “Smart” build infrastructure



5.83% of build failures  
BUT  
only due to 16 configuration  
bugs of Linux and 3  
configuration bugs of... TUXML

**We come to this insight thanks  
to our learning procedure**





# Embedded Linux Conference Europe