

Large-scale Experiments with the Configuration Space of the Linux Kernel (XuG talk)



@acherm

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

Preprints (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>



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.
```

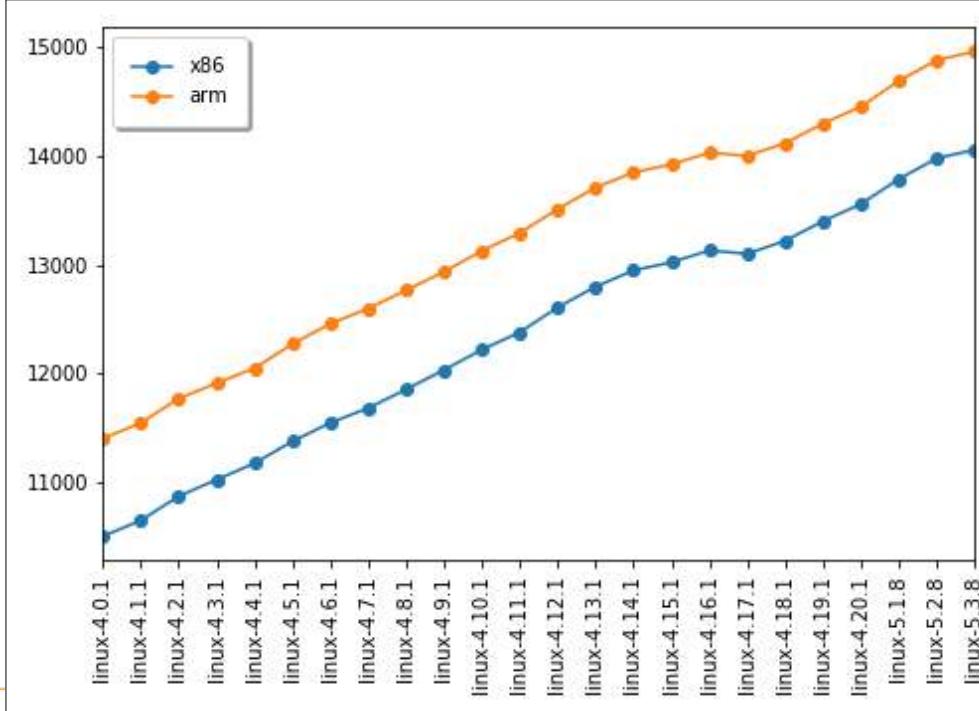
This allows 32-bit apic IDs (so it can support very large systems),

```
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_MP_PARSE=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
```



15,000+
options



Configurations:
Hell or Heaven?

Stop ou encore?

`..../linux-5.8`

`make`



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

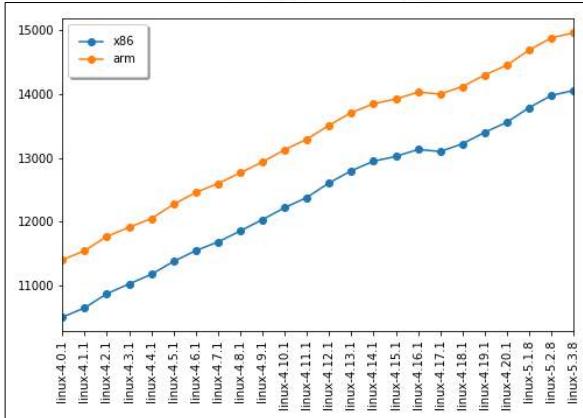
```
# Processor type and features
#
# CONFIG_ZONE_DMA is not set
# CONFIG_SMP is not set
# CONFIG_X86_FEATURE_NAMES is not set
# CONFIG_X86_FAST_FEATURE_TESTS is not set
CONFIG_X86_X2APIC=y
CONFIG_X86_MP_PARSE=y
CONFIG_GOLDFISH=y
# CONFIG_INTEL_RDT_A is not set
# CONFIG_X86_EXTENDED_PLATFORM is not set
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
BOOL	36.40
INT	1.54
STRING	0.29
HEX	0.14

39000
26000

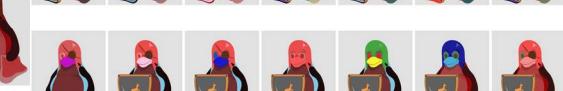
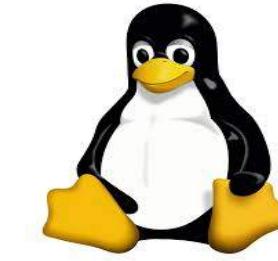
Linux 5.2.8, arm
(% of types' options)

$\approx 10^{6000}$ configurations
(without constraints)

Linux Kernel

$\approx 10^{6000}$

configurations





Linux Kernel
... **$\approx 10^{60}$** configurations

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

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

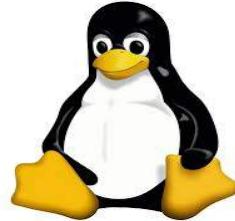


Linux vs AlphaZero

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

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
```

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_WO_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_SHUFFLE_PAGE_ALLOCATOR=y
CONFIG_SLUB_CACHING=y
CONFIG_SLUB_ALLOCATION=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_WO_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_MP_PARSE=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_WO_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

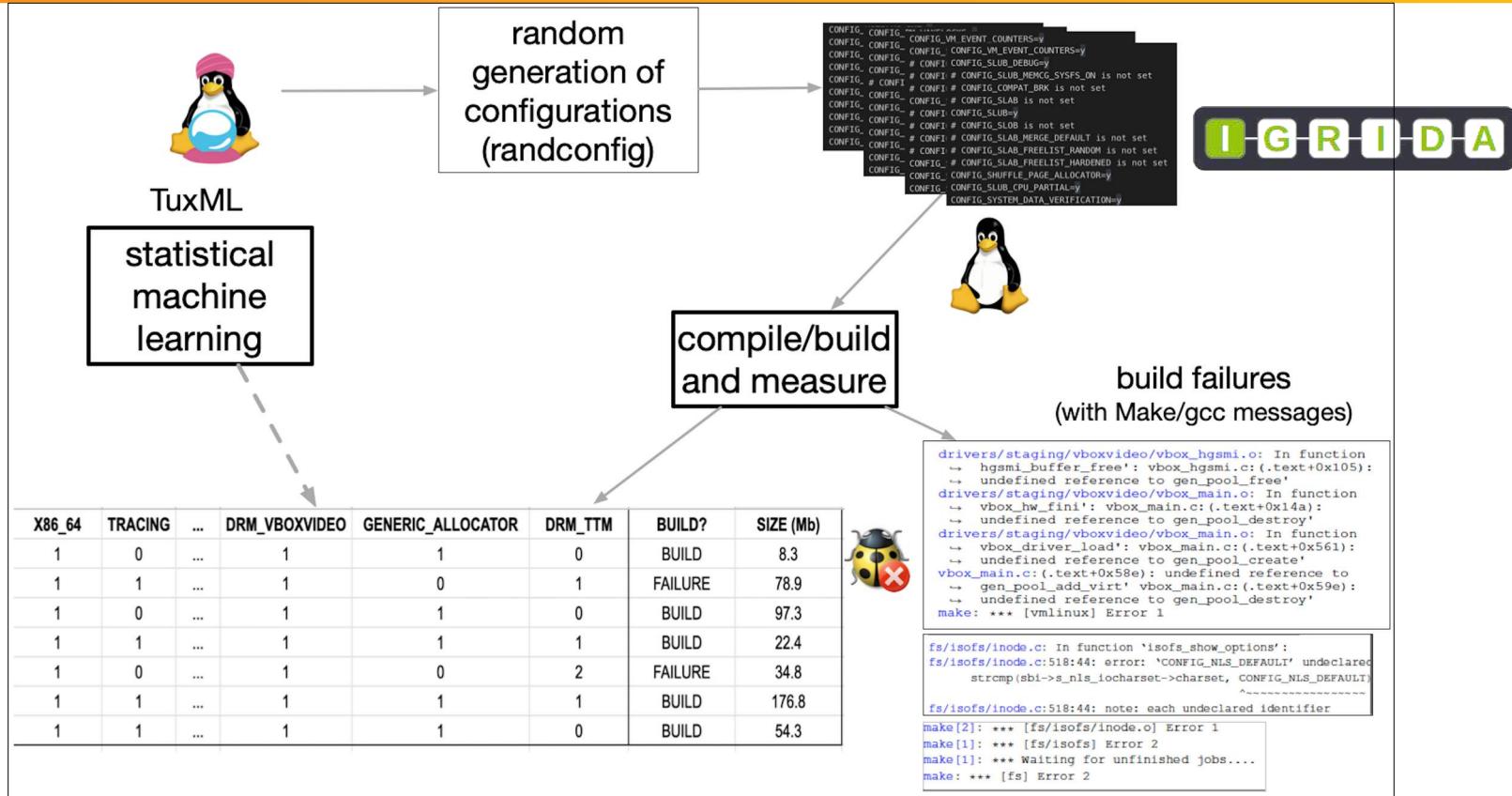
```
CONFIG_PM_WAKELOCKS=y
CONFIG_PM_WAKELOCKS_LIMIT=100
CONFIG_PM_WAKELOCKS_GC=y
CONFIG_PM=y
# CONFIG_PM_DEBUG is not set
CONFIG_PM_CLK=y
CONFIG_PM_GENERIC_DOMAINS=y
CONFIG_WO_POWER_EFFICIENT_DEFAULT=y
CONFIG_PM_GENERIC_DOMAINS_SLEEP=y
CONFIG_PM_GENERIC_DOMAINS_OF=y
CONFIG_ENERGY_MODEL=y
CONFIG_ARCH_SUPPORTS_ACPI=y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

102.3Mb



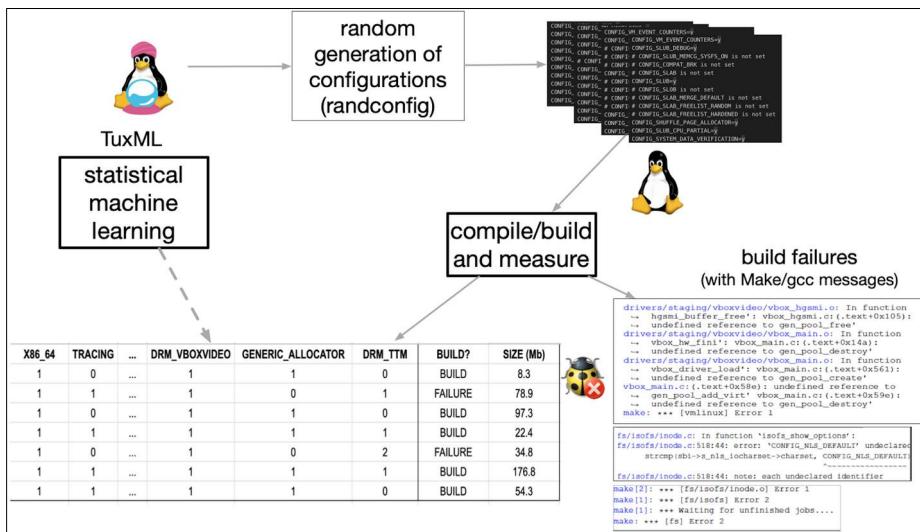
Regression problem:
predict a quantitative value (eg size)
out of options values

TUXML: Sampling, Measuring, Learning



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 ▼ 1	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 ...

(information sent to a database)

Data: arch x86_64, gcc 6.4

cid	compilation_date	compilation_time	config_file	stdout_log_file	stderr_log_file	user_output_file	compiled_kernel_size	compressed_compiled_kernel_size
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 KiB]	[BLOB - 33.7 KiB]	[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 KiB]	[BLOB - 34.7 KiB]	[BLOB - 14 o]	[BLOB - 704 o]	81377768	GZIP-bzImage : 18375632 , GZIP-vmlinux : 20462408 ...
74880	2019-08-12 16:47:28	301.775	[BLOB - 22 KiB]	[BLOB - 24.2 KiB]	[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 KiB]	[BLOB - 50 KiB]	[BLOB - 571 o]	[BLOB - 712 o]	109098328	GZIP-bzImage : 17183792 , GZIP-vmlinux : 19272160 ...
74878	2019-08-10 14:54:03	305.705	[BLOB - 26.1 KiB]	[BLOB - 28.8 KiB]	[BLOB - 14 o]	[BLOB - 703 o]	55523752	GZIP-bzImage : 14767568 , GZIP-vmlinux : 16862088 ...

Kernel version	Release date	Number of configurations	Seconds/config
4.13	2017/09/03	92,562	not available
4.15	2018/01/28	39,391	not available
4.2	2018/12/23	23,489	225
5	2019/03/03	19,952	247
5.4	2019/10/24	25,847	285
5.7	2020/05/31	20,159	258
5.8	2020/08/02	21,923	289

(15K hours of computation on a grid computing for 4.13.3)



Application 1: “Smart” build infrastructure

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 KiB]	[BLOB - 33.7 KiB]	[BLOB - 14.0 KiB]	[BLOB - 702.0 KiB]	74559280	GZIP-bzImage : 8865504 , GZIP-vmlinux : 10943304 ...
74881	2019-08-12 16:58:09	460.392	[BLOB - 25.8 KiB]	[BLOB - 34.7 KiB]	[BLOB - 14.0 KiB]	[BLOB - 704.0 KiB]	81377768	GZIP-bzImage : 18375632 , GZIP-vmlinux : 20462408 ...
74880	2019-08-12 16:47:28	301.775	[BLOB - 22 KiB]	[BLOB - 24.2 KiB]	[BLOB - 14.0 KiB]	[BLOB - 705.0 KiB]	83004496	GZIP-bzImage : 14365648 , GZIP-vmlinux : 18452424 ...
74879	2019-08-12 16:46:14	1393.61	[BLOB - 24.1 KiB]	[BLOB - 50 KiB]	[BLOB - 571.0 KiB]	[BLOB - 712.0 KiB]	109096328	GZIP-bzImage : 17183792 , GZIP-vmlinux : 19272160 ...
Console de requêtes SQL	2019-08-12 16:46:03	305.705	[BLOB - 26.1 KiB]	[BLOB - 28.8 KiB]	[BLOB - 14.0 KiB]	[BLOB - 703.0 KiB]	55523752	GZIP-bzImage : 14767568 , GZIP-vmlinux - 16 seconds

results for 4.13.3 only



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?



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_RELAXED_ALIGNMENT=y
ELIST_RANDOM is not set
ELIST_HARDENED is not set
CONFIG_ALLOCATOR=y
ARTIAL=y
A_VERIFICATION=y
```

X

build failing

```
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



A classification tree diagram showing the decision path for various Linux configuration options. The root node is `CONFIG_PM_WAKELOCKS=y`. The tree splits into two main paths based on the value of `CONFIG_PM_WAKELOCKS`.

```

graph TD
    Root[CONFIG_PM_WAKELOCKS=y] --> Node1[AIC79XX_BUILD_FIRMWARE <= 0.5]
    Node1 -- True --> Node2[WANXL_BUILD_FIRMWARE <= 0.5]
    Node1 -- False --> Node3[GENERIC_ALLOCATOR <= 0.5]
    Node2 --> Node4[DRM_VBOXVIDEO <= 1.5]
    Node2 --> Node5[UBSAN_SANITIZE_ALL <= 0.5]
    Node3 --> Node6[VIDEO_XILINX_VTC <= 0.5]
  
```

The leaf nodes provide details on Gini impurity, sample size, and configuration values.

Configuration Option	Gini	Samples	Value
<code>AIC79XX_BUILD_FIRMWARE</code>	0.073	86486	<code>[3256, 83228]</code>
<code>WANXL_BUILD_FIRMWARE</code>	0.02	85023	<code>[1795, 83228]</code>
<code>GENERIC_ALLOCATOR</code>	0.011	83699	<code>[471, 83228]</code>
<code>DRM_VBOXVIDEO</code>	0.162	4695	<code>[344, 4351]</code>
<code>UBSAN_SANITIZE_ALL</code>	0.003	79004	<code>[127, 78877]</code>
<code>VIDEO_XILINX_VTC</code>	0.002	74419	<code>[58, 74361]</code>

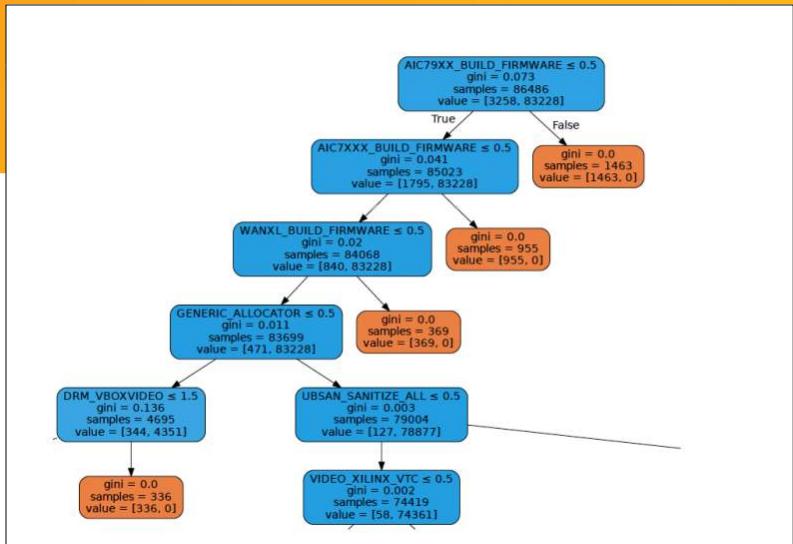
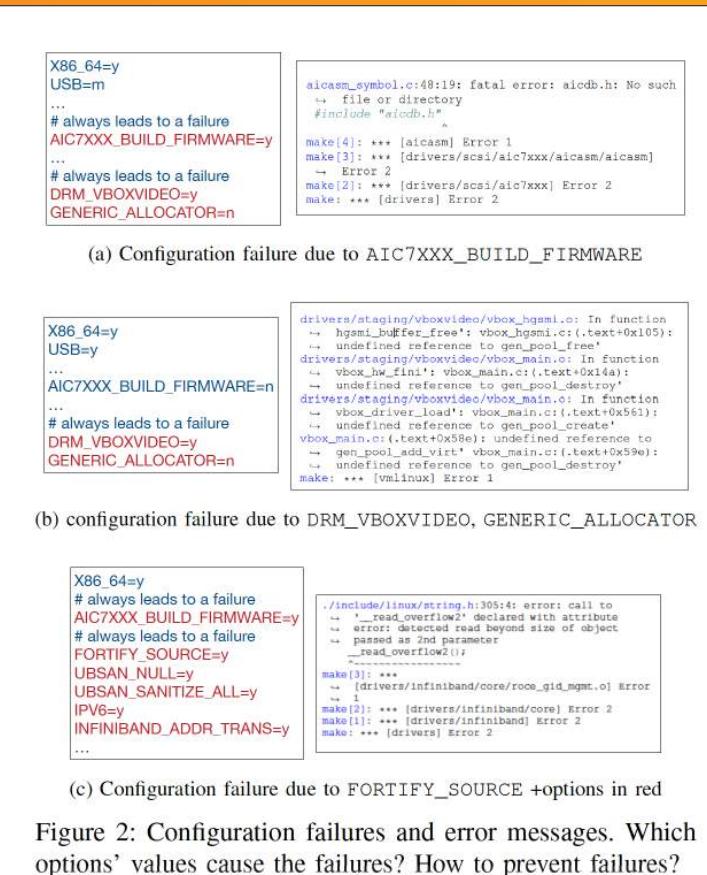
On the right, there are two examples of classification results:

- Passing Build:** Shows a green bar and the text "build passing".
- Failing Build:** Shows a red X over a yellow bar and the text "build failing".

Configuration Option	Build?	Size (Mb)
<code>GENERIC_ALLOCATOR</code>	0	8.3
<code>DRM_TTM</code>	1	78.9
<code>GENERIC_ALLOCATOR</code>	1	97.3
<code>DRM_TTM</code>	1	22.4
<code>GENERIC_ALLOCATOR</code>	2	34.8
<code>DRM_TTM</code>	1	176.8
<code>GENERIC_ALLOCATOR</code>	0	54.3

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

Application 1: “Smart” build



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

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

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

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_G07007 & 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_MP_PARSE=y
CONFIG_GOLDFISH=y
# CONFIG_INTEL_RST_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_PM_GENERIC_DOMAINS_OF=y
CONFIG_SYSTEM_DATA_VERIFICATION=y
```

16.1Mb

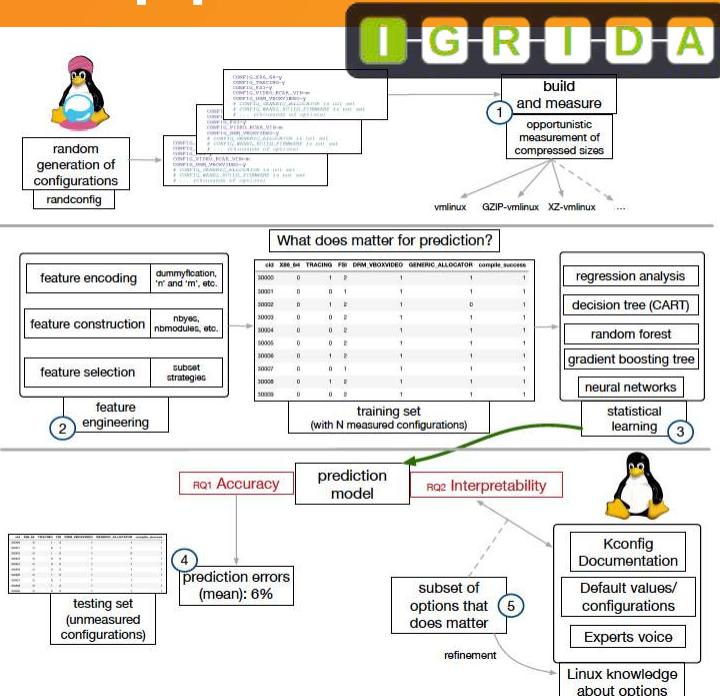
```
CONFIG_PM_WAKELOCKS=y
CONFIG_PM_WAKELOCKS_LIMIT=100
CONFIG_PM_WAKELOCKS_GC=y
CONFIG_PM=y
# CONFIG_PM_DEBUG is not set
CONFIG_PM_CLK=y
CONFIG_PM_GENERIC_DOMAINS=y
CONFIG_WQ_POWER_EFFICIENT_DEFAULT=y
CONFIG_PM_GENERIC_DOMAINS_SLEEP=y
CONFIG_PM_GENERIC_DOMAINS_OF=y
CONFIG_ENERGY_MODEL=y
CONFIG_ARCH_SUPPORTS_ACPI=y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

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



(size of vmlinux)

Max: 1,698.14Mb

Min: 7Mb (tinyconfig)

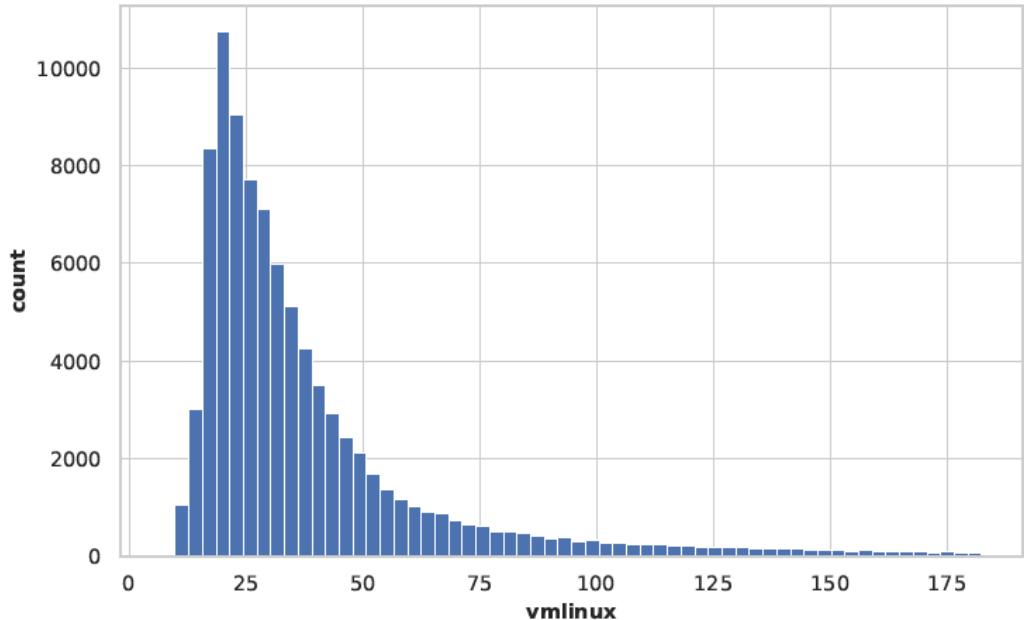


Figure 4: Distribution of size (in Mb) without outliers

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)

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

Conclusion 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

Challenges (XuG brainstorm!)

- Docker for reproducible build (+ because many tools are needed to build *any* random configs)
 - Variability paranoia: we want to control the computational environment... gcc ≥ 9 has been a challenging surprise!
 - Is Docker a good solution?
- How to measure compilation/boot time?



Challenges (XuG brainstorm!)

- The cost of gathering data is important
 - Idea: incremental build of configurations
- The computation of kernel properties is hard to instrument
 - Boot: physical boot or QEMU?
 - Boot time: accurate measure?
 - Security: is a config. safe wrt Spectre/Meltdown?
 - Energy consumption: boot+bench; accuracy?



Challenges (XuG brainstorm!)

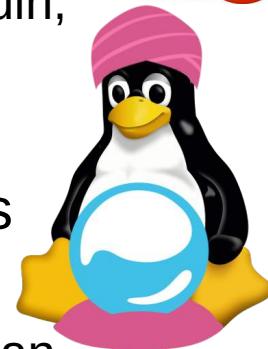
- Database: right now a VM managed at ISTIC
 - Manual backups
 - I would like to have an institutional-supported solution (but public/security/admin issues)
- Collaborations with KernelCI
 - Moving towards cloud-based solutions?



 KernelCI is now a [Linux Foundation project](#) supported by:

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 Barai
 - <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 11th 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 Dumangé, 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!)

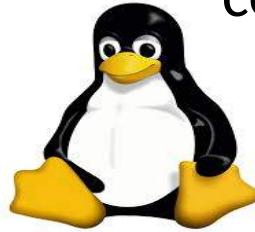


Intrigued by Tux logos?

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

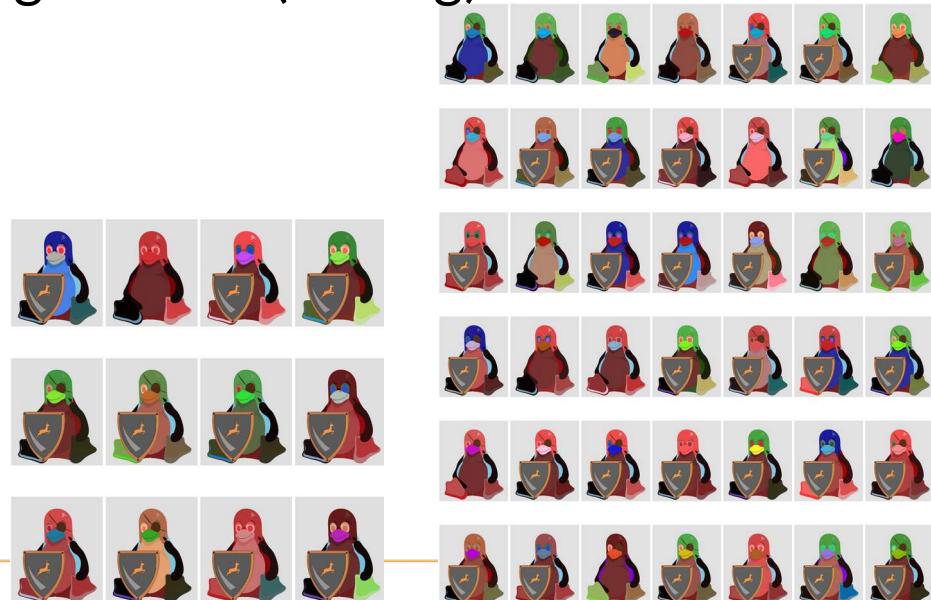
Eliot Marie

Pierre Pouteau

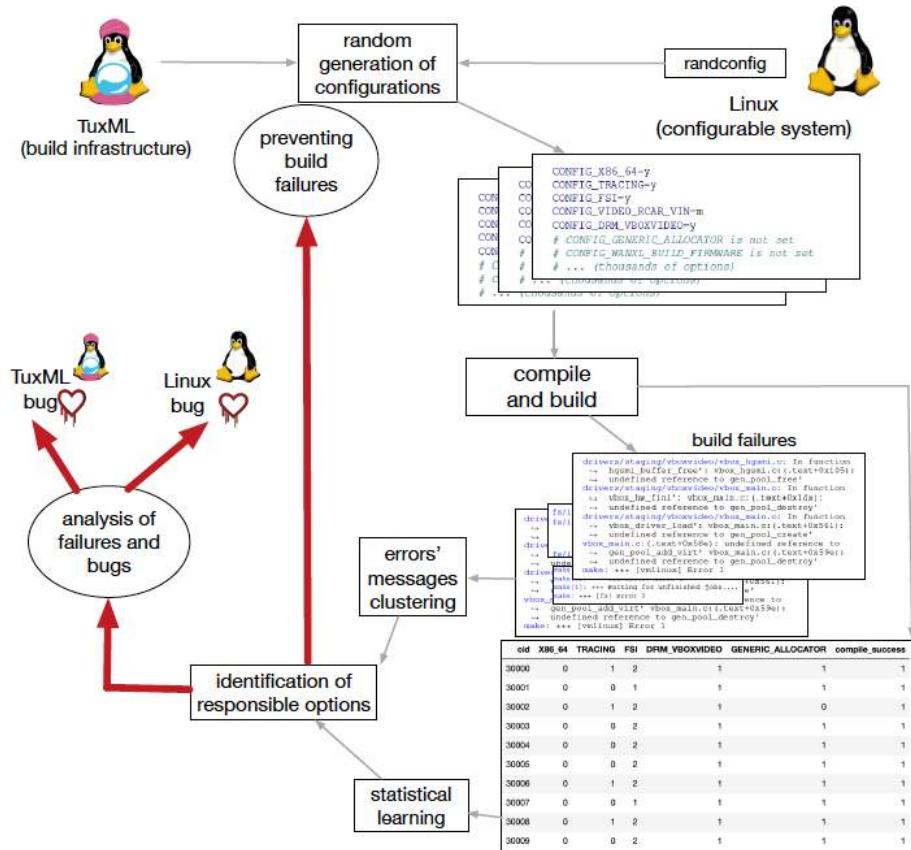
Zakariae Boukhchen

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

