Feature Consistency in Compile-Time–Configurable System Software

Facing the Linux 10000 Feature Problem

Reinhard Tartler, Daniel Lohmann, Julio Sincero,
Wolfgang Schröder-Preikschat

System Software Group

Friedrich-Alexander University
Erlangen-Nuremberg

September 7, 2011

supported by DFG

Configuration Complexity

- Linux has become incredibly configurable

Configuration Complexity

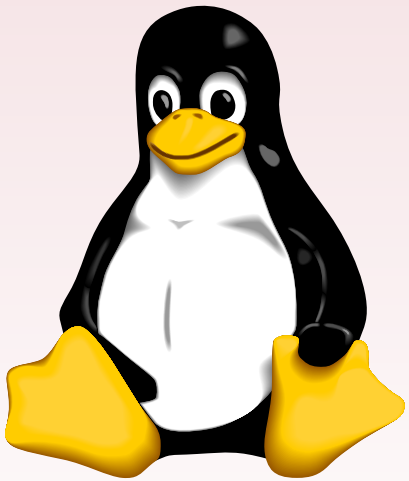
- Linux has become incredibly configurable
- Complexity increases considerably

Configuration Complexity

- Linux has become incredibly configurable
- Complexity increases considerably

⇝ Source of **bugs**!

Configuration Complexity



Linux v3.0 contains:

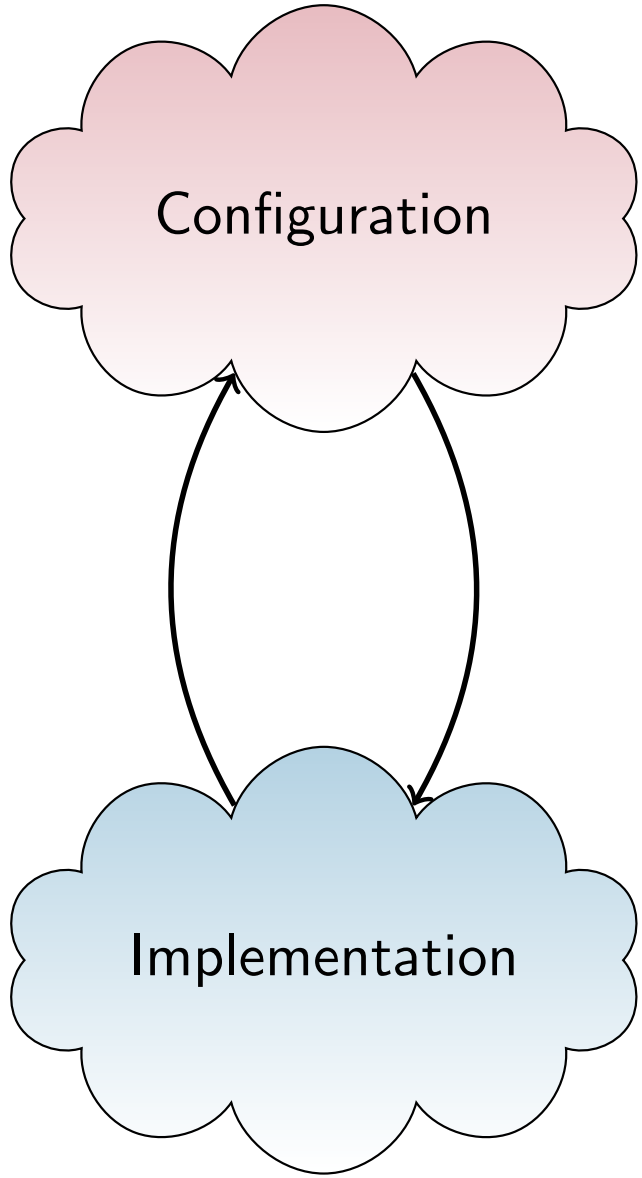
7.702 Features

893 Kconfig files

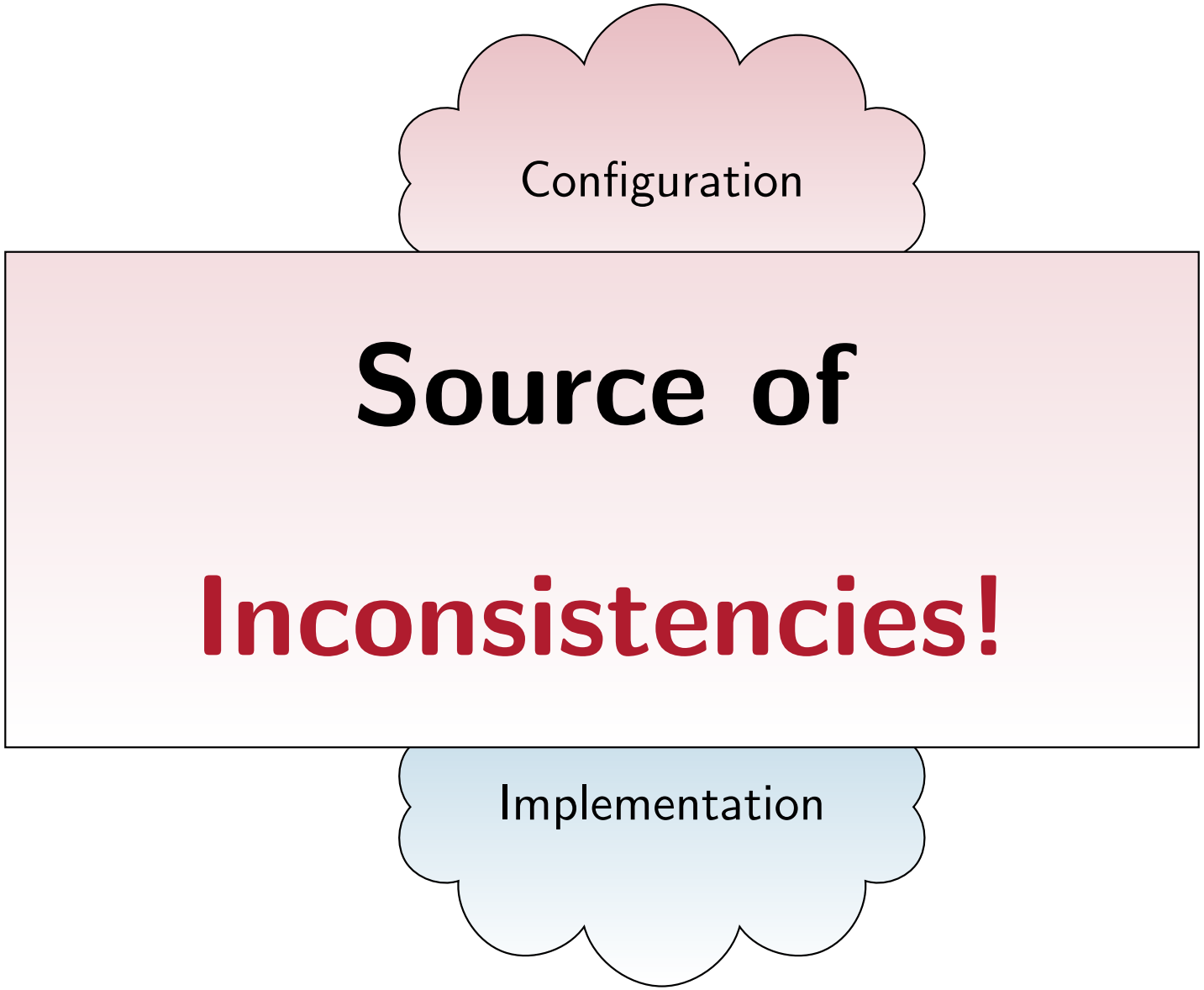
31.281 Source files

88.897 `#ifdef` blocks

The Problem



The Problem

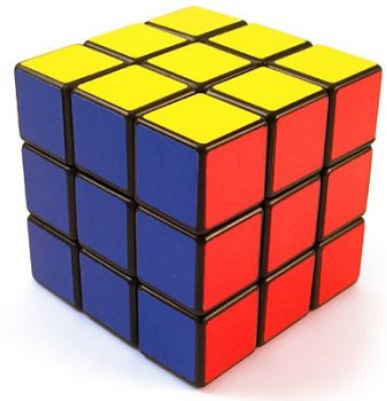

Configuration
Source of
Inconsistencies!
Implementation

Finding Bugs with Tools for Static Analysis

- Bugs in declaration and implementation
- Excellent tool support for static analysis:
  - Coccinelle: Faults in Linux: Ten Years Later (ASPLOS'11)
  - Dingo: Taming Device Drivers (EuroSys'09)
  - KLEE: Automatic generation of high-coverage tests (EuroSys'08)
  - RWset: Attacking path explosion (TACAS'08)
  - EXE: Automatically generating inputs of death (CCS'06)
  - ...

Finding Bugs with Tools for Static Analysis

- Bugs in declaration and implementation
- Excellent tool support for static analysis:
  - Coccinelle: Faults in Linux: Ten Years Later (ASPLOS'11)
  - Dingo: Taming Device Drivers (EuroSys'09)
  - KLEE: Automatic generation of high-coverage tests (EuroSys'08)
  - RWset: Attacking path explosion (TACAS'08)
  - EXE: Automatically generating inputs of death (CCS'06)
  - ...
- Each of them check a single configuration:



Symbolic Inconsistency

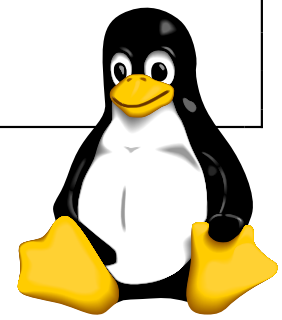
```
config HOTPLUG_CPU
        bool "Support for hot-pluggable CPUs"
        depends on SMP && HOTPLUG
        ---help---
```

Symbolic Inconsistency

```
config HOTPLUG_CPU
        bool "Support for hot-pluggable CPUs"
        depends on SMP && HOTPLUG
        ---help---
```

```
static int
hotplug_cfd(struct notifier_block *nfb, unsigned long action, void *hcpu)
{
        // [...]
        switch (action) {
        case CPU_UP_PREPARE:
        case CPU_UP_PREPARE_FROZEN:
                // [...]
#ifdef CONFIG_CPU_HOTPLUG
        case CPU_UP_CANCELED:
        case CPU_UP_CANCELED_FROZEN:

        case CPU_DEAD:
        case CPU_DEAD_FROZEN:
                free_cpumask_var(cfd->cpumask);
                break;
#endif
        };
        return NOTIFY_OK;
```



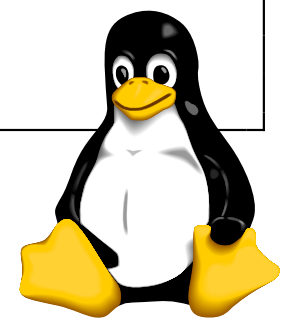
Symbolic Inconsistency

```
config HOTPLUG_CPU
        bool "Support for hot-pluggable CPUs"
        depends on SMP && HOTPLUG
        ---help---
```

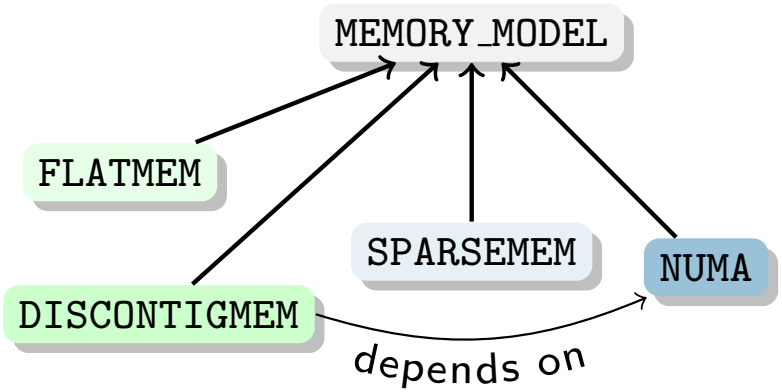
```
static int
hotplug_cfd(struct notifier_block *nfb, unsigned long action, void *hcpu)
{
        // [...]
        switch (action) {
        case CPU_UP_PREPARE:
        case CPU_UP_PREPARE_FROZEN:
                // [...]
#ifdef CONFIG_CPU_HOTPLUG
        case CPU_UP_CANCELED:
        case CPU_UP_CANCELED_FROZEN:

        case CPU_DEAD:
        case CPU_DEAD_FROZEN:
                free_cpumask_var(cfd->cpumask);
                break;
#endif
        };
        return NOTIFY_OK;
```

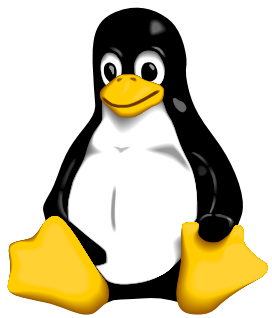
- Result: Fix for a critical bug



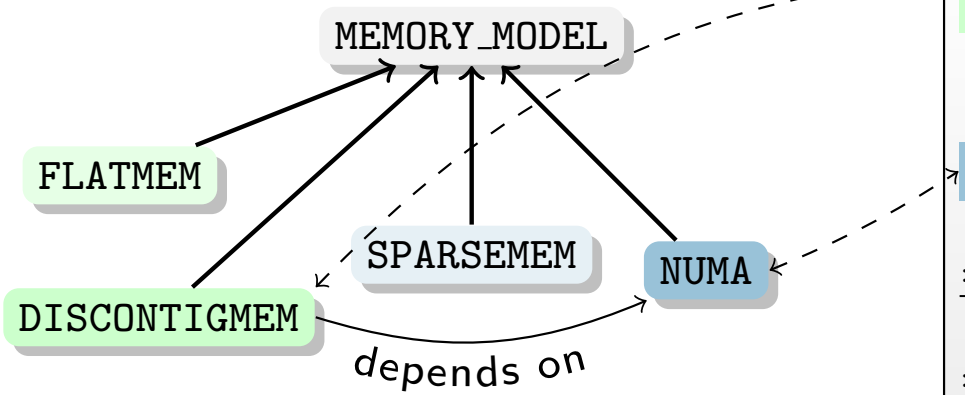
Logic Inconsistencies




```
#ifdef CONFIG_DISCONTIGMEM
  // Block1
  static ... int pfn_to_mid(...)
# ifdef CONFIG_NUMA
    // Block2
# else
    // Block3
# endif
#endif
```

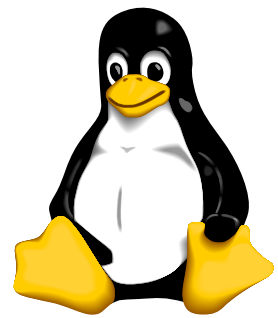


Logic Inconsistencies

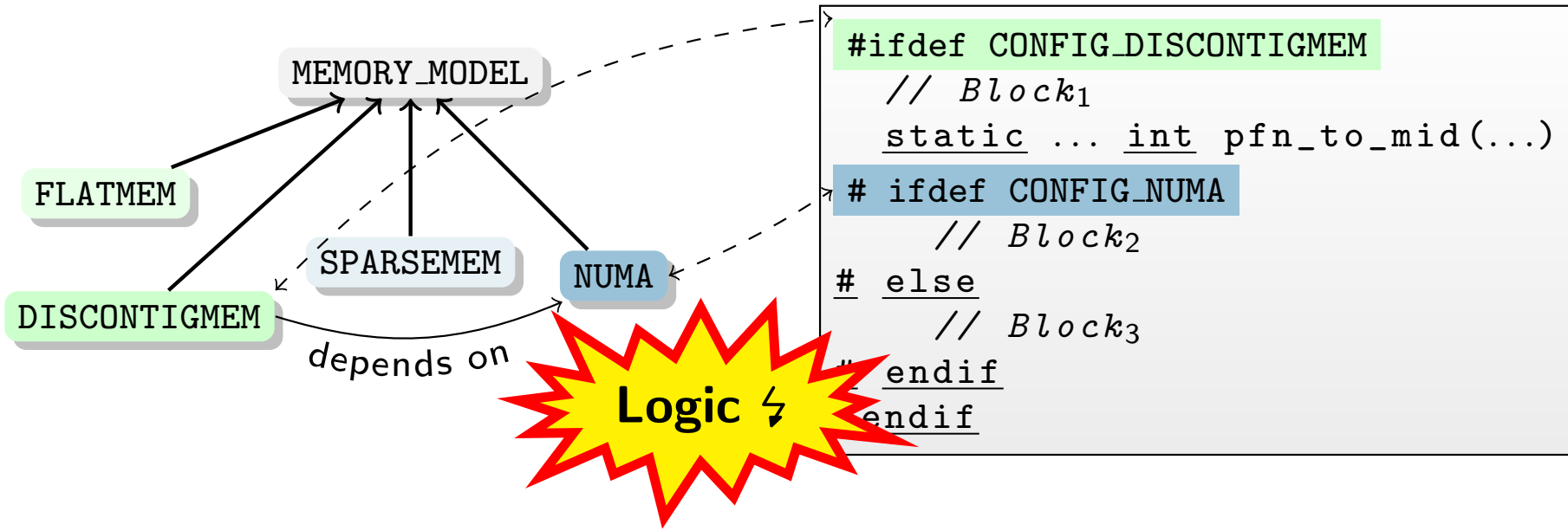



```
#ifdef CONFIG_DISCONTIGMEM
  // Block1
  static ... int pfn_to_mid(...)
# ifdef CONFIG_NUMA
    // Block2
# else
    // Block3
# endif
#endif
```

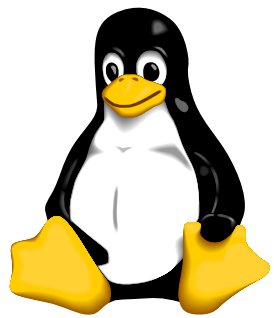
- Feature DISCONTIGMEM requires NUMA
- Inner block is not configuration dependent anymore



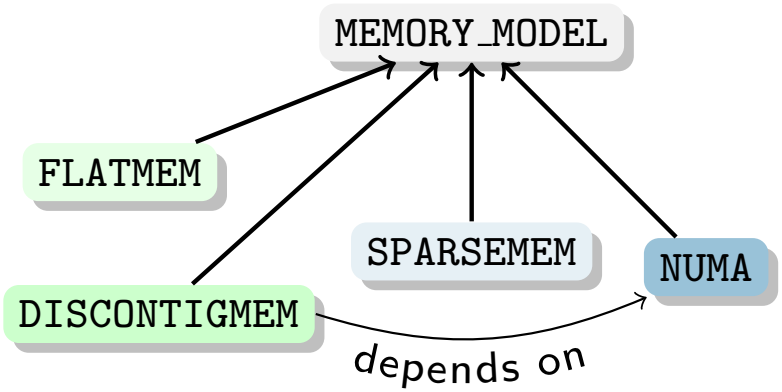
Logic Inconsistencies



- Feature DISCONTIGMEM requires NUMA
- Inner block is not configuration dependent anymore
- Result: code cleanup

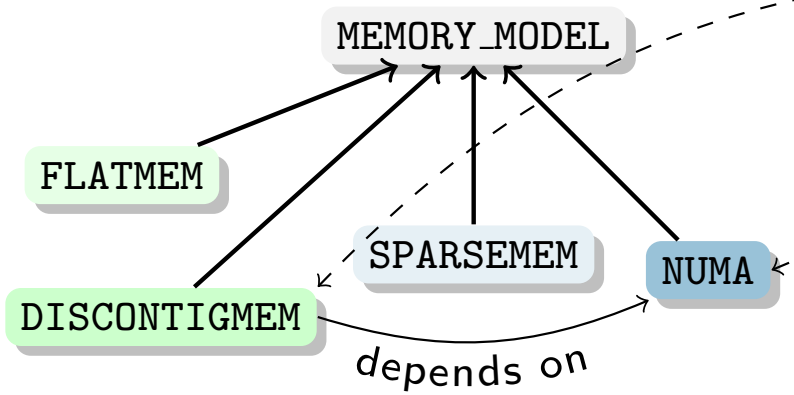


General Approach



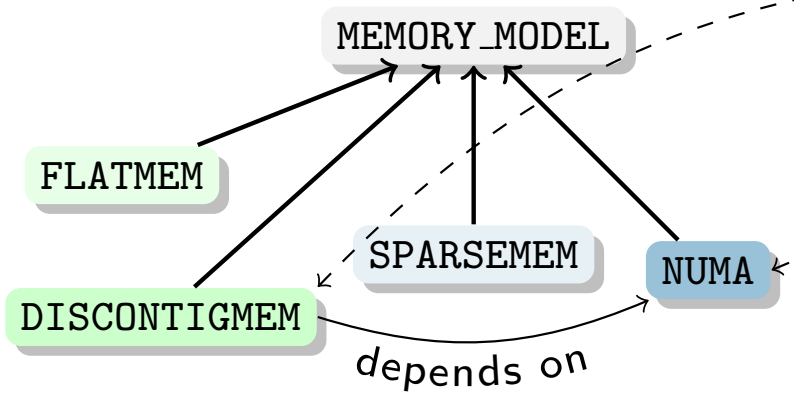

```
#ifdef CONFIG_DISCONTIGMEM
  // Block1
  static ... int pfn_to_mid(...)
# ifdef CONFIG_NUMA
    // Block2
# else
    // Block3
# endif
#endif
```

General Approach




```
#ifdef CONFIG_DISCONTIGMEM
   // Block1
   static ... int pfn_to_mid(...)
# ifdef CONFIG_NUMA
      // Block2
# else
      // Block3
# endif
#endif
```

General Approach

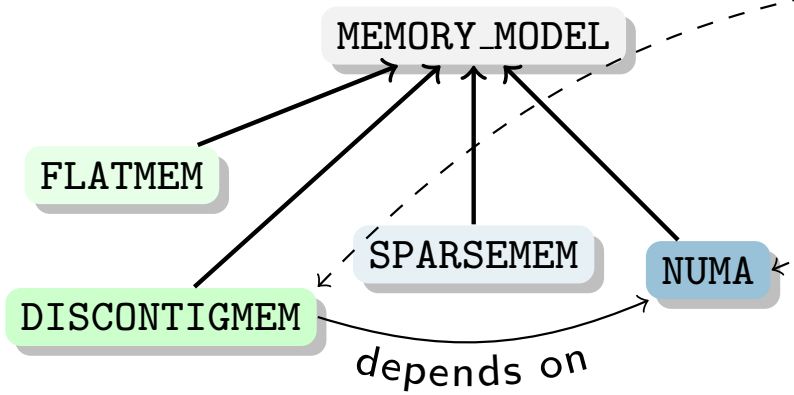



```
#ifdef CONFIG_DISCONTIGMEM
   // Block1
   static ... int pfn_to_mid(...)
# ifdef CONFIG_NUMA
      // Block2
# else
      // Block3
# endif
#endif
```

extract

$$\begin{aligned}\mathcal{C} = \quad & (\texttt{FLATMEM} \rightarrow \texttt{MEMORY\_MODEL}) \\ \wedge\ & (\texttt{DISCONTIGMEM} \rightarrow \texttt{MEMORY\_MODEL}) \\ \wedge\ & (\texttt{SPARSEMEM} \rightarrow \texttt{MEMORY\_MODEL}) \\ \wedge\ & (\texttt{NUMA} \rightarrow \texttt{MEMORY\_MODEL}) \\ \wedge\ & (\texttt{DISCONTIGMEM} \rightarrow \texttt{NUMA})\end{aligned}$$

General Approach




```
#ifdef CONFIG_DISCONTIGMEM
   // Block1
   static ... int pfn_to_mid(...)
# ifdef CONFIG_NUMA
      // Block2
#  else
      // Block3
#  endif
#endif
```

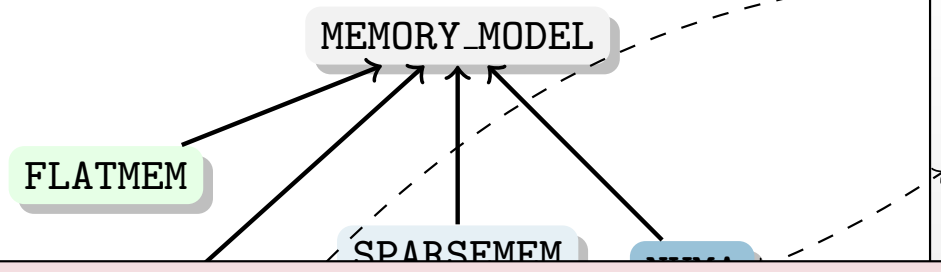
extract

$$\mathcal{C} = (\texttt{FLATMEM} \rightarrow \texttt{MEMORY\_MODEL}) \wedge (\texttt{DISCONTIGMEM} \rightarrow \texttt{MEMORY\_MODEL}) \wedge (\texttt{SPARSEMEM} \rightarrow \texttt{MEMORY\_MODEL}) \wedge (\texttt{NUMA} \rightarrow \texttt{MEMORY\_MODEL}) \wedge (\texttt{DISCONTIGMEM} \rightarrow \texttt{NUMA})$$

extract

$$\mathcal{I} = (\texttt{Block}_1 \leftrightarrow \texttt{DISCONTIGMEM}) \wedge (\texttt{Block}_2 \leftrightarrow \texttt{Block}_1 \wedge (\texttt{NUMA}) \wedge (\texttt{Block}_3 \leftrightarrow \texttt{Block}_1 \wedge \neg\texttt{Block}_2)$$

General Approach




```
#ifdef CONFIG_DISCONTIGMEM
  // Block1
  static ... int pfn_to_mid(...)
# ifdef CONFIG_NUMA
    // Block2
```

Crosscheck both formulas with a SAT solver:

$$\textbf{dead?} \quad = \quad sat(\mathcal{C} \wedge \mathcal{I} \wedge \mathtt{Block}_N)$$

$$\textbf{undead?} \quad = \quad sat(\mathcal{C} \wedge \mathcal{I} \wedge \neg\mathtt{Block}_N \wedge \ parent(\mathtt{Block}_N))$$

$$\wedge\,(\mathtt{DISCONTIGMEM} \rightarrow \mathtt{MEMORY\_MODEL})$$
$$\wedge\,(\mathtt{SPARSEMEM} \rightarrow \mathtt{MEMORY\_MODEL})$$
$$\wedge\,(\mathtt{NUMA} \rightarrow \mathtt{MEMORY\_MODEL})$$
$$\wedge\,(\mathtt{DISCONTIGMEM} \rightarrow \mathtt{NUMA})$$

M)

$$\wedge\,(\mathtt{Block}_2 \leftrightarrow \mathtt{Block}_1 \wedge (\mathtt{NUMA})$$
$$\wedge\,(\mathtt{Block}_3 \leftrightarrow \mathtt{Block}_1 \wedge \neg\mathtt{Block}_2)$$

Implementation Challenges

Accuracy

- Conceptually no false positives
- Exact identification of variation points

Implementation Challenges

- Accuracy
  - Conceptually no false positives
  - Exact identification of variation points
- Coverage
  - Extract configuration model for all **22** architectures
  - Defect $\leadsto$ detected on each architecture

Implementation Challenges

- Accuracy
  - Conceptually no false positives
  - Exact identification of variation points
- Coverage
  - Extract configuration model for all **22** architectures
  - Defect $\leadsto$ detected on each architecture
- Performance
  - Easy and fast to use during incremental builds
  - Possible by problem slicing
  - Complete run on Linux in less than **10** minutes

Results

subsystem	#ifdefs	logic	symbolic	total
arch/	33757	345	581	926
drivers/	32695	88	648	736
fs/	3000	4	13	17
include/	7241	6	11	17
kernel/	1412	7	2	9
mm/	555	0	1	1
net/	2731	1	49	50
sound/	3246	5	10	15
virt/	53	0	0	0
other subsystems	601	4	1	5
$\sum$	85291	460	1316	1776
fix proposed		150 (1)	214 (22)	364 (23)
confirmed defect		38 (1)	116 (20)	154 (21)
confirmed rule-violation		88 (0)	21 (2)	109 (2)
pending		24 (0)	77 (0)	101 (0)

Results

We have found **1776** configurability issues

Submitted **123** patches for **364** defects

20 are confirmed **new bugs** (affecting binary code)

Cleaned up **5129** lines of cruft code

Further Application: Configuration Coverage

- Current ongoing work, accepted at PLOS'11
- Configuration Coverage is defined as:

  fraction of *selected* configuration-conditional blocks
  divided by the number of *available* configuration-conditional blocks.

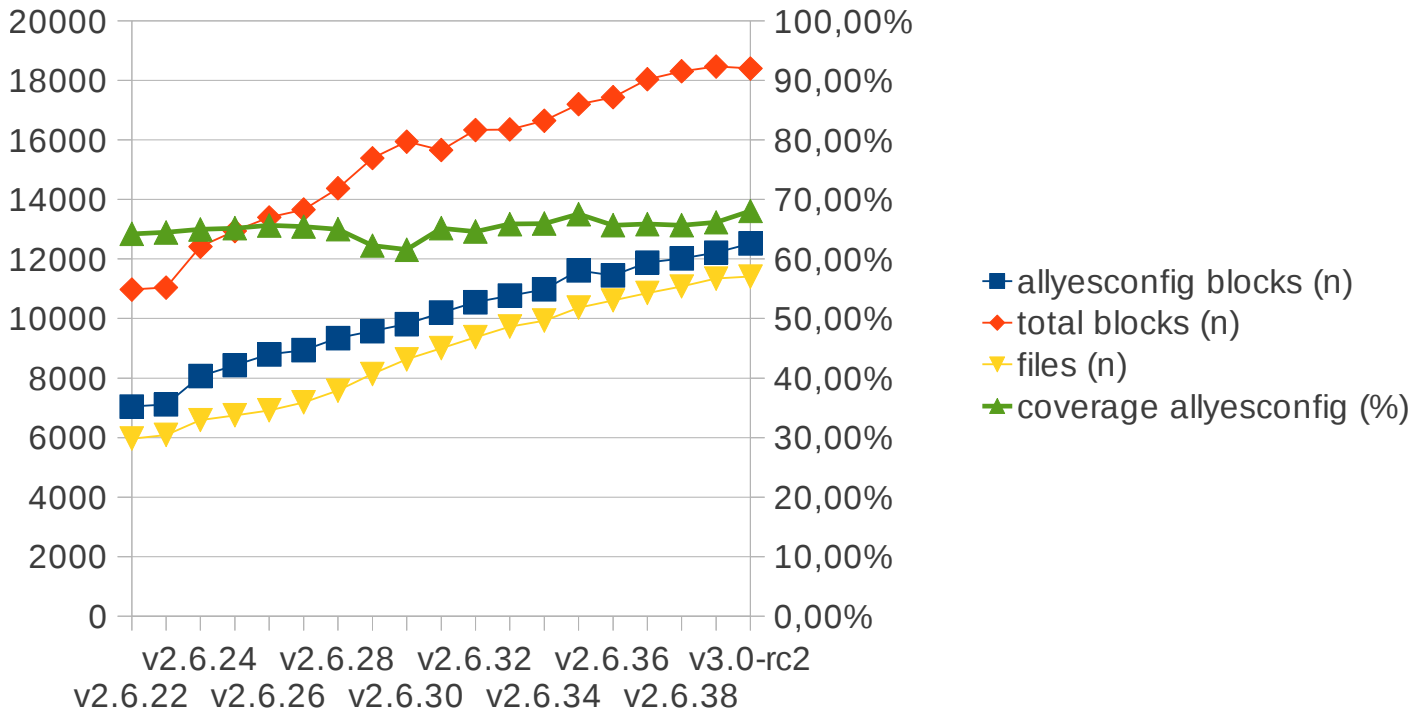
- How to catch bugs that apply only on specific kernel configurations?

  $\Rightarrow$ Test them on as many configurations as possible

- Static analyzers (sparse, smatch, ...) scan a particular kernel configuration

  $\Rightarrow$ How to *effeciently* exand their coverage?

Historical analysis of `allyes` coverage

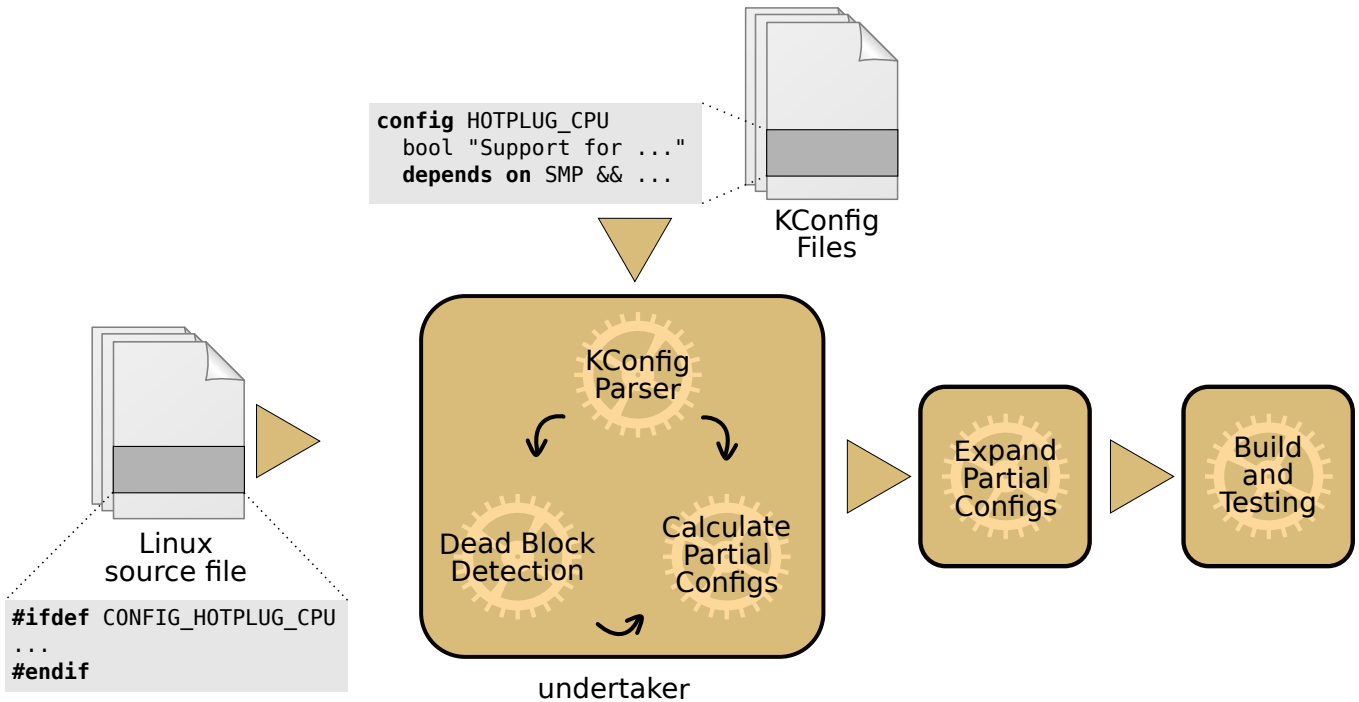


Concrete Example

```
#ifdef CONFIG_DISCONTIGMEM                                        Block 1
static inline int pfn_to_nid(unsigned long pfn)
{
#ifdef CONFIG_NUMA                                                Block 2
   return((int) physnode_map[(pfn) / PAGES_PER_ELEMENT]);
#else                                                             Block 3
   return 0;
#endif
}
#endif                                                            Block 1
```

- Possible Configurations:
  - Neither, `DISCONTIGMEM`, `DISCONTIGMEM` $\wedge$ `NUMA`
- Additionally testing the configuration `NUMA` does not increase the Configuration Coverage.

Realization



Implementation Challenges

- Proper extraction of Configurations constraints
  - Kconfig (implemented in undertaker)
  - Kbuild constraints (largely unhandled)
- Expansion of Partial Configurations
  - Naïve approach has some surprising effects (i.e., fails sometimes)
  - Kconfig-sat seems promising, but unfortunately discontinued

Evaluation

Analyzed files	10,365
Files with variability	3,163
Rate of files with variability	30.52%
Sum of all (partial) configurations	4,435
Sum of *configuration controlled* conditional blocks	16,444
Sum of blocks selected by `allyesconfig`	11,511
Sum of all blocks selected by `undertaker-coverage`	13,844
Coverage `allyesconfig` (non-dead-corrected)	70.00%
Coverage `undertaker` (non-dead-corrected)	84.19%
Dead blocks	1,778
Selectable blocks (excluding dead blocks)	14,666
Selected by `allyesconfig`	11,511
Covered by `undertaker`	13,844
`allyesconfig` coverage	78.49%
`undertaker` coverage	94.40%
`undertaker` coverage / `allyesconfig` coverage	1.20

Evaluation

Analyzed files	10,365
Files with variability	**3,163**
Rate of files with variability	30.52%
Sum of all (partial) configurations	**4,435**
Sum of *configuration controlled* conditional blocks	16,444
Sum of blocks selected by `allyesconfig`	11,511
Sum of all blocks selected by `undertaker-coverage`	13,844
Coverage `allyesconfig` (non-dead-corrected)	70.00%
Coverage `undertaker` (non-dead-corrected)	84.19%
Dead blocks	1,778
Selectable blocks (excluding dead blocks)	14,666
Selected by `allyesconfig`	11,511
Covered by `undertaker`	13,844
`allyesconfig` coverage	78.49%
`undertaker` coverage	94.40%
`undertaker` coverage / `allyesconfig` coverage	1.20

Evaluation

Analyzed files	10,365
Files with variability	**3,163**
Rate of files with variability	30.52%
Sum of all (partial) configurations	**4,435**
Sum of *configuration controlled* conditional blocks	16,444
Sum of blocks selected by `allyesconfig`	11,511
Sum of all blocks selected by `undertaker-coverage`	13,844
Coverage `allyesconfig` (non-dead-corrected)	**70.00%**
Coverage `undertaker` (non-dead-corrected)	**84.19%**
Dead blocks	1,778
Selectable blocks (excluding dead blocks)	14,666
Selected by `allyesconfig`	11,511
Covered by `undertaker`	13,844
`allyesconfig` coverage	78.49%
`undertaker` coverage	94.40%
`undertaker` coverage / `allyesconfig` coverage	1.20

Evaluation

Analyzed files	10,365
Files with variability	**3,163**
Rate of files with variability	30.52%
Sum of all (partial) configurations	**4,435**
Sum of *configuration controlled* conditional blocks	16,444
Sum of blocks selected by `allyesconfig`	11,511
Sum of all blocks selected by `undertaker-coverage`	13,844
Coverage `allyesconfig` (non-dead-corrected)	**70.00%**
Coverage `undertaker` (non-dead-corrected)	**84.19%**
Dead blocks	1,778
Selectable blocks (excluding dead blocks)	14,666
Selected by `allyesconfig`	11,511
Covered by `undertaker`	13,844
`allyesconfig` **coverage**	**78.49%**
`undertaker` **coverage**	**94.40%**
`undertaker` coverage / `allyesconfig` coverage	1.20

Evaluation

Analyzed files	10,365
Files with variability	**3,163**
Rate of files with variability	30.52%
Sum of all (partial) configurations	**4,435**

With **30 percent** more compiler calls
(static analysis runs)

We get **15 percent** more Configuration Coverage

Covered by `undertaker`	13,844
`allyesconfig` **coverage**	**78.49%**
`undertaker` **coverage**	**94.40%**
`undertaker` coverage / `allyesconfig` coverage	1.20

Conclusions

- Configurability has to be seen as a significant cause of software defects in its own respect
- Configuration and implementation need to be kept consistent
- Configuration Coverage increases the effectiveness of existing tools.

Conclusions

- Configurability has to be seen as a significant cause of software defects in its own respect
- Configuration and implementation need to be kept consistent
- Configuration Coverage increases the effectiveness of existing tools.
- Vision:
  - Explorative tool for visualizing and checking Variability in Kconfig and realization
  - Linux Feature Explorer (LIFE)

Conclusions

- Configurability has to be seen as a significant cause of software defects in its own respect
- Configuration and implementation need to be kept consistent
- Configuration Coverage increases the effectiveness of existing tools.
- Vision:
  - Explorative tool for visualizing and checking Variability in Kconfig and realization
  - Linux Feature Explorer (LIFE)

`http://vamos.informatik.uni-erlangen.de/trac/undertaker`