Where are my knobs?

Unveiling Hidden Configuration Knobs and Investigating their Implications on Performance

Lukas Abelt

Florian Sattler

Sven Apel

Saarland Informatics Campus
Saarland University

FOSD Meeting 2025, Köthen

Hidden Configurability?

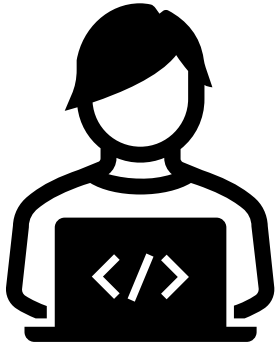
- Performance depends on configuration
- Can we access all options?
  →**Hidden** configuration opportunities?
- What is “Hidden”?
  - It depends…

```
void foo() {
  const int boost_threshold = 500;

  // ...

  if (count > boost_threshold) {
    fast_implementation()
  } else {
    slow_implementation()
  }
}
```

Hidden Configurability
Roles



Developers

- **Build** configurable SW systems
- Make **design** decisions:
  - Select configuration options
  - Specify ranges for options
  - Provide documentation

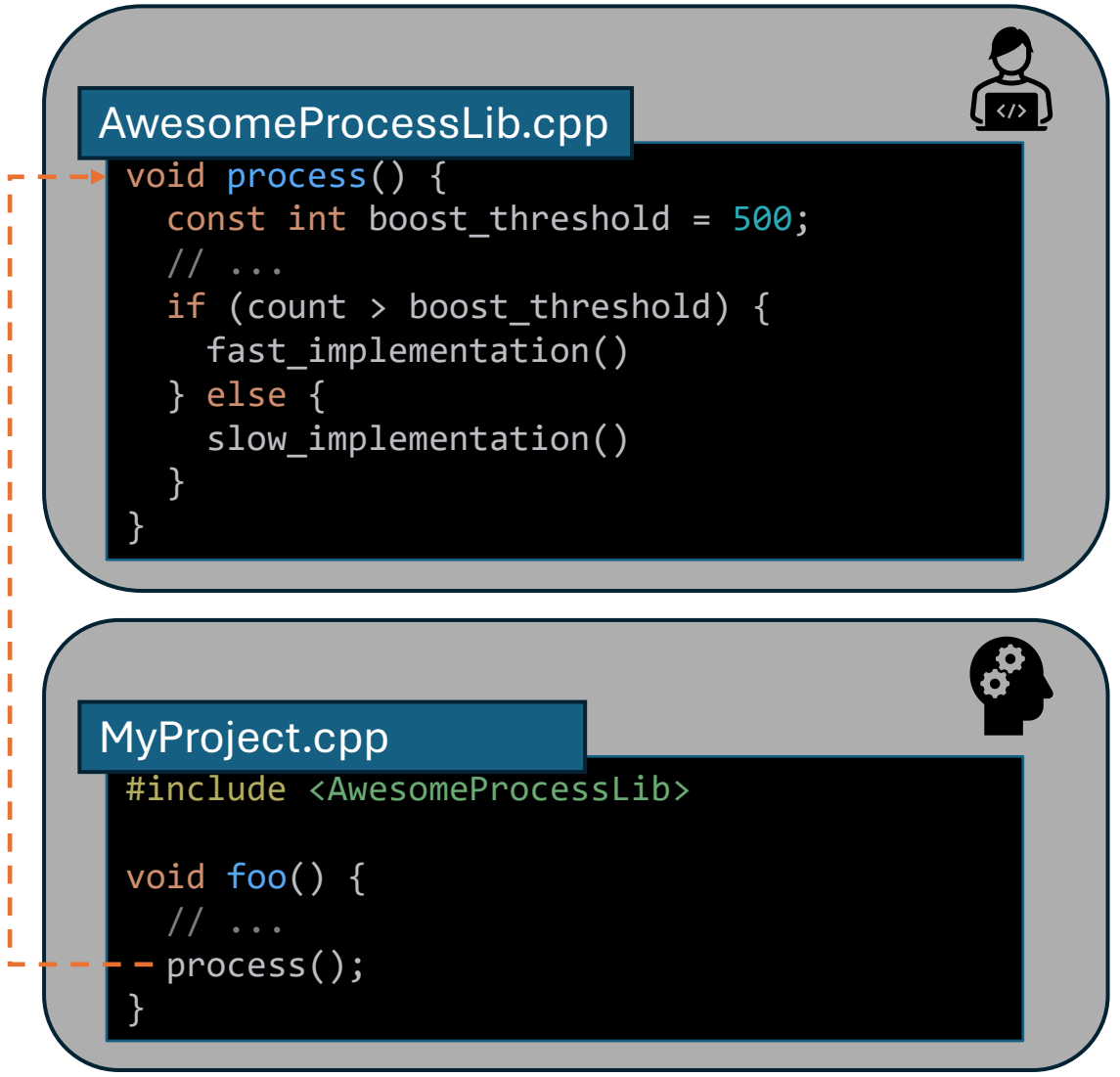
Clients

- **Use** configurable SW systems
- Make **configuration** decisions:
  - Identify relevant options
  - Explore configuration range
  - Select suitable values for use-case

Hidden Configurability
Caused by Developers

Configuration Opportunities

- Code that may serve as configuration knob
- Configuration may be beneficial
  - For some clients
- **Cannot** be configured by the clients
  - Except with developer code changes



Hidden Configurability
Caused by Clients

Unrealized Configurability

- Configuration knob is there
- But not set, due to:
  - Visibility
  - Unawareness
  - Lack of domain knowledge
  - Default values
  - Missing documentation
- Lead to **uninformed** decision

Hidden Configurability

Goals

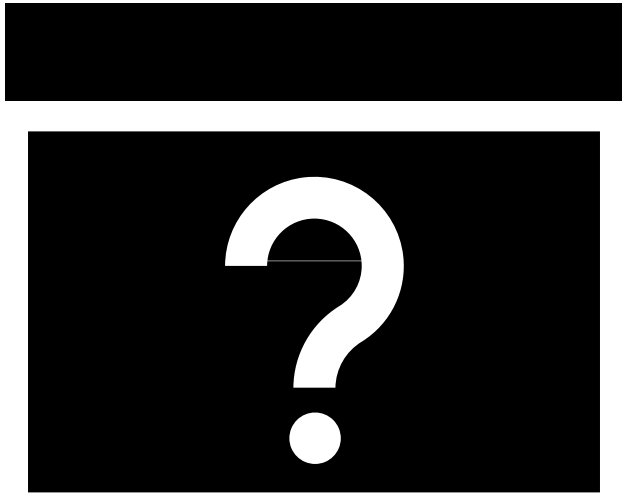
1. Finding Hidden Configurability
   - Automatic
   - Light-weight

2. Impact of Hidden Configurability
   - Performance Effects

3. Understanding Hidden Configurability
   - Why is it there?
   - How can we prevent it?
   - Can we “fix” it?

Hidden Configurability

Pipeline



Hidden Configurability

Pipeline – Step 1: Identification

Hidden Configurability Candidate Patterns ™

```
traverse(
    varDecl(allOf(
    hasType(hasUnqualifiedDesugaredType(builtinType())),
    hasType(isConstQualified()),
    hasInitializer(AnyLiteralType)));
```

Open-Source Projects

VP9 ØMQ bzip2 Dune

Source Code

Hidden Configurability Detector Tool ™

AST Matcher

Candidate Locations

```
- Locations:
    - File: "src/dune/common/basic.cpp"
      Lineno: 25
      Colno: 5
    - File: "src/dune/common/grid.cpp"
      Lineno: 42
      Colno: 3
```

Hidden Configurability

Pipeline – Step 2: Classification

Project	Locations
BROTLI	8
DUNE	303
FASTDOWNWARD	14
HYTEG	1097
LIBVPX	321
LIBZMQ	50

Manual Classification

?

Ideas:
- History Information
- LLMs?

```
static float tansig_approx(float x) {
  const float N0 = 952.52801514f;
  const float N1 = 96.39235687f;
```

```
bool ParseVpxCodecPrivate(const uint8_t*
data, Vp9CodecFeatures* features) {
  const int kVpxCodecPrivateMinLength = 3;
  const uint8_t kVp9ProfileId = 1;
```

```
// Is 1000 a sensible default?
const int default_hwm = 1000;
```

Configuration Opportunity!

Now

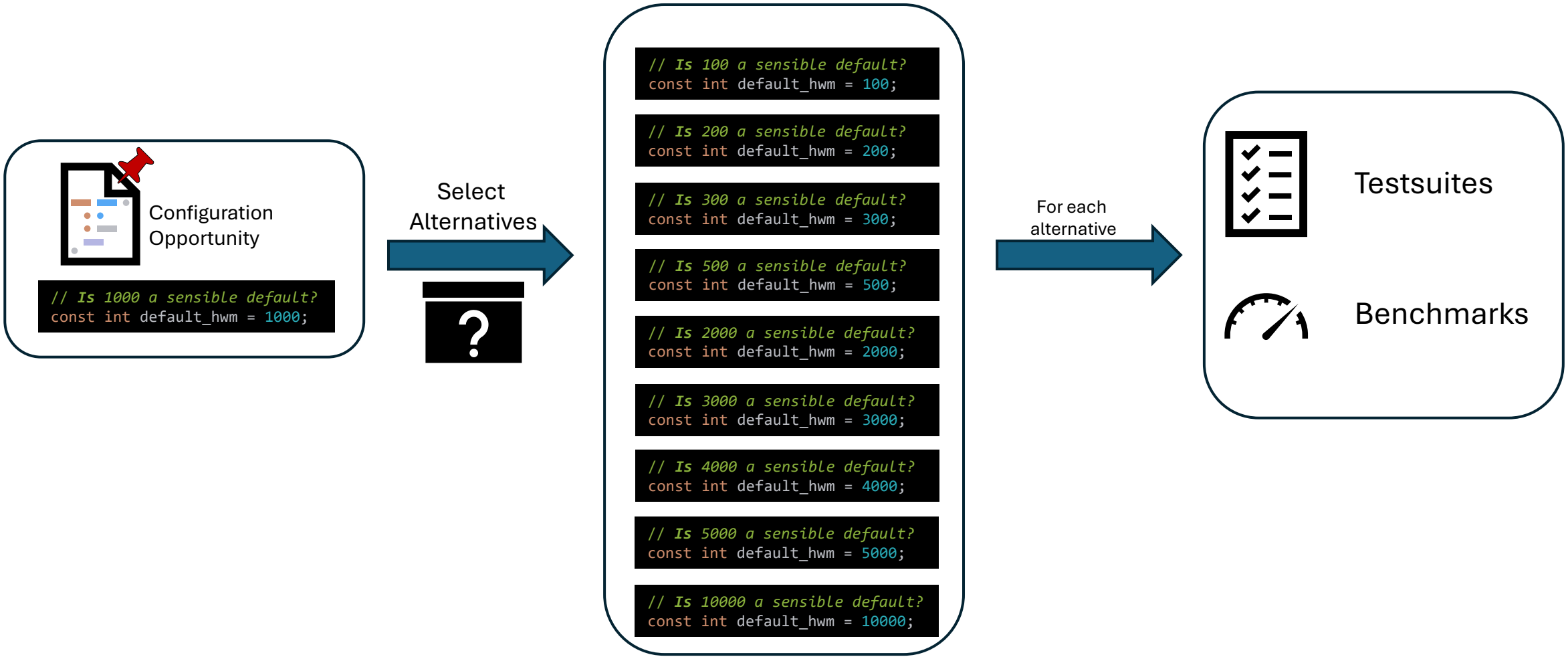
```
const double kLiteralBlockSwitchCost = 28.1;
const double kCommandBlockSwitchCost = 13.5;
const double kDistanceBlockSwitchCost = 14.6;
```

2014

```
const double kLiteralBlockSwitchCost = 26;
const double kCommandBlockSwitchCost = 13.5;
const double kDistanceBlockSwitchCost = 14.6;
```

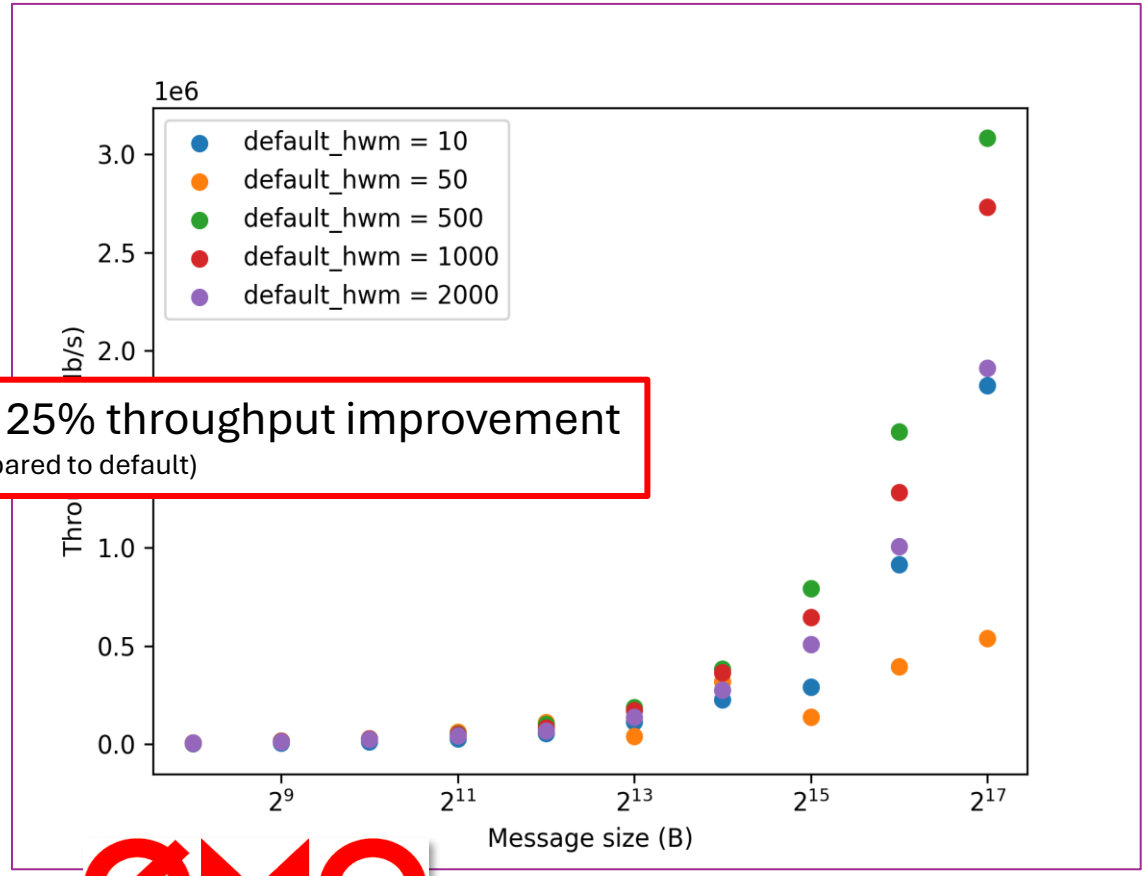
Hidden Configurability

Pipeline – Step 3: Variation



Hidden Configurability

Insights



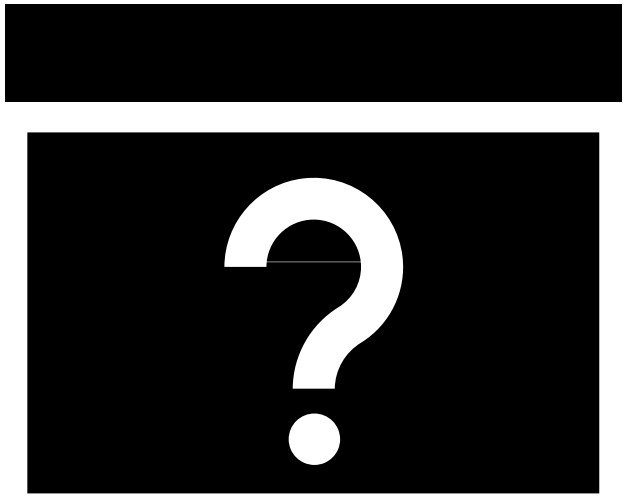
Up to 25% throughput improvement

(As compared to default)

ØMQ

```
// Is 1000 a sensible default?
const int default_hwm = 1000;
```

Hidden Configurability
Summary

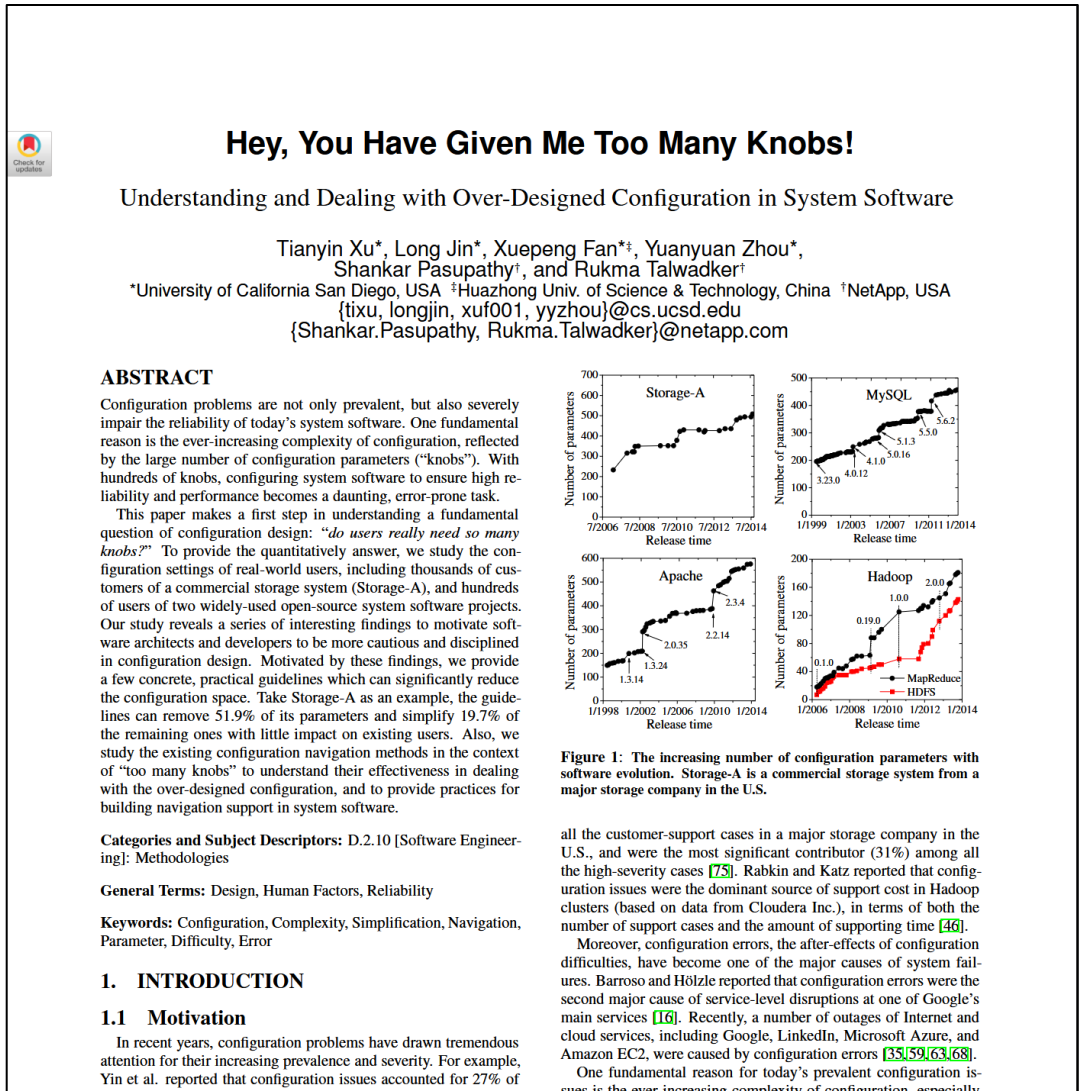


- **How** to:
  1. Select Candidate Patterns?
  2. Filter Candidates?
  3. Select Alternatives?

Closing Remarks

Are we the baddies?

- Tl;dr:
  - There are already too many options to understand
- Do we need more features?



Hey, You Have Given Me Too Many Knobs!

Understanding and Dealing with Over-Designed Configuration in System Software

Tianyin Xu*, Long Jin*, Xuepeng Fan*‡, Yuanyuan Zhou*,
Shankar Pasupathy†, and Rukma Talwadker†
*University of California San Diego, USA ‡Huazhong Univ. of Science & Technology, China †NetApp, USA
{tixu, longjin, xuf001, yyzhou}@cs.ucsd.edu
{Shankar.Pasupathy, Rukma.Talwadker}@netapp.com

ABSTRACT

Configuration problems are not only prevalent, but also severely impair the reliability of today's system software. One fundamental reason is the ever-increasing complexity of configuration, reflected by the large number of configuration parameters ("knobs"). With hundreds of knobs, configuring system software to ensure high reliability and performance becomes a daunting, error-prone task.

This paper makes a first step in understanding a fundamental question of configuration design: "*do users really need so many knobs?*" To provide the quantitatively answer, we study the configuration settings of real-world users, including thousands of customers of a commercial storage system (Storage-A), and hundreds of users of two widely-used open-source system software projects. Our study reveals a series of interesting findings to motivate software architects and developers to be more cautious and disciplined in configuration design. Motivated by these findings, we provide a few concrete, practical guidelines which can significantly reduce the configuration space. Take Storage-A as an example, the guidelines can remove 51.9% of its parameters and simplify 19.7% of the remaining ones with little impact on existing users. Also, we study the existing configuration navigation methods in the context of "too many knobs" to understand their effectiveness in dealing with the over-designed configuration, and to provide practices for building navigation support in system software.

Categories and Subject Descriptors: D.2.10 [Software Engineering]: Methodologies

General Terms: Design, Human Factors, Reliability

Keywords: Configuration, Complexity, Simplification, Navigation, Parameter, Difficulty, Error

1. INTRODUCTION

1.1 Motivation

In recent years, configuration problems have drawn tremendous attention for their increasing prevalence and severity. For example, Yin et al. reported that configuration issues accounted for 27% of

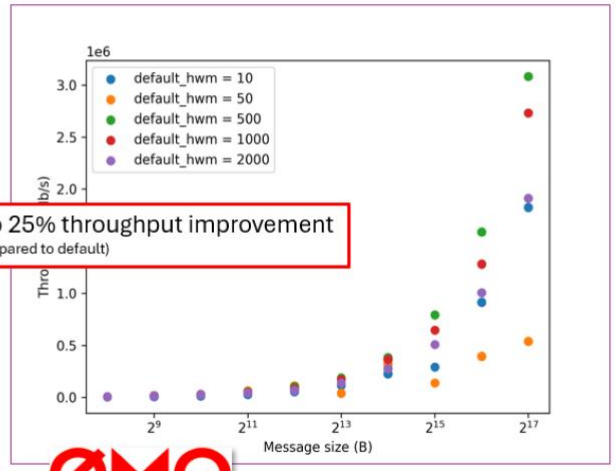
Figure 1: The increasing number of configuration parameters with software evolution. Storage-A is a commercial storage system from a major storage company in the U.S.

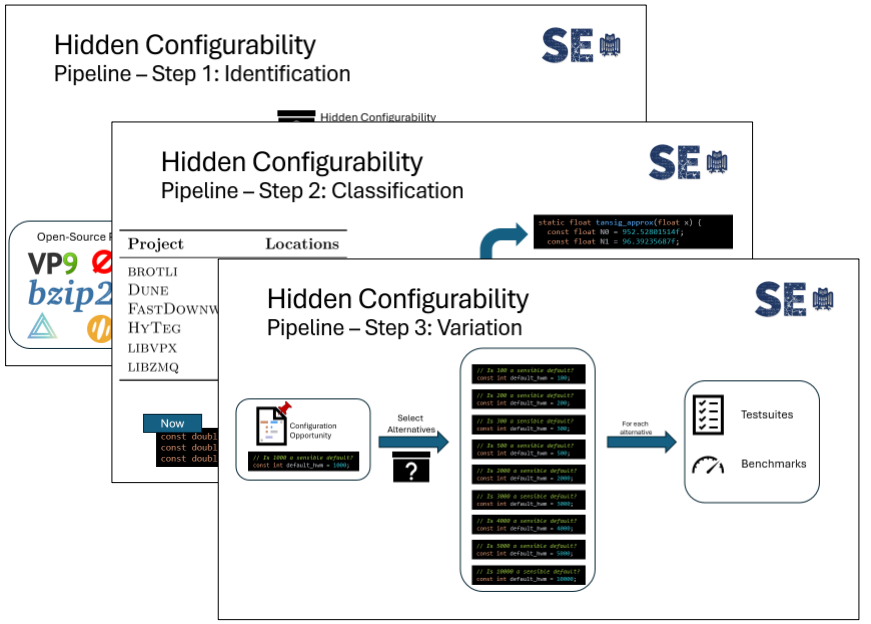
all the customer-support cases in a major storage company in the U.S., and were the most significant contributor (31%) among all the high-severity cases [75]. Rabkin and Katz reported that configuration issues were the dominant source of support cost in Hadoop clusters (based on data from Cloudera Inc.), in terms of both the number of support cases and the amount of supporting time [46].

Moreover, configuration errors, the after-effects of configuration difficulties, have become one of the major causes of system failures. Barroso and Hölzle reported that configuration errors were the second major cause of service-level disruptions at one of Google's main services [16]. Recently, a number of outages of Internet and cloud services, including Google, LinkedIn, Microsoft Azure, and Amazon EC2, were caused by configuration errors [35, 59, 63, 68].

One fundamental reason for today's prevalent configuration is-

Hidden Configurability
Insights





Lukas Abelt
abeltluk@cs.uni-saarland.de

Saarland Informatics Campus
Saarland University
FOSD Meeting 2025, Köthen

Project	Total Locations	Filtered Locations
BROTLI	8	5
DUNE	303	7
FASTDOWNWARD	14	5
HYTEG	1097	0
LIBVPX	321	9
LIBZMQ	50	1