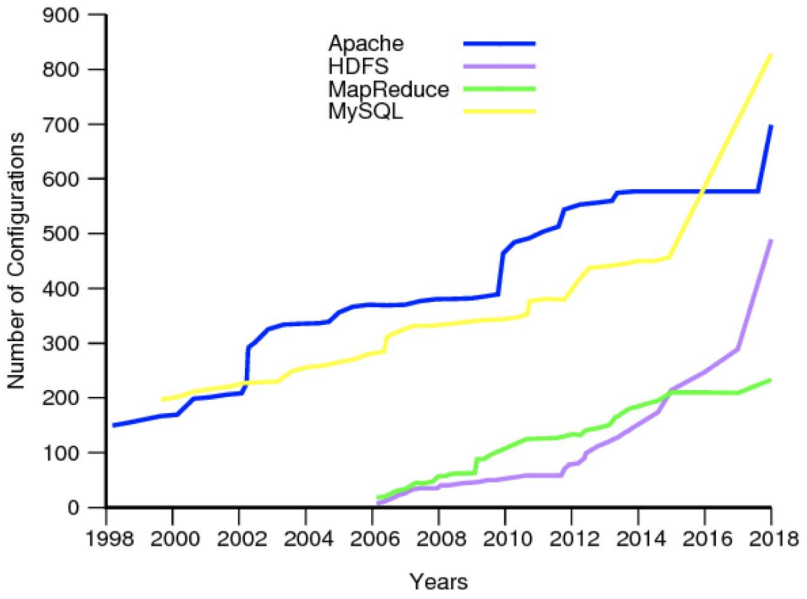
An Evolutionary Study of Configuration Design and Implementation in Cloud Systems

Yuanliang Zhang[1,2], Haochen He[1], Owolabi Legunsen[3], Shanshan Li[1],
Wei Dong[1], Tianyin Xu[2]

[1]National University of Defense Technology
[2]University of Illinois at Urbana-Champaign
[3]Cornell University

Motivation



Complexity increases rapidly over time

Previous: misconfiguration detecting and diagnosing

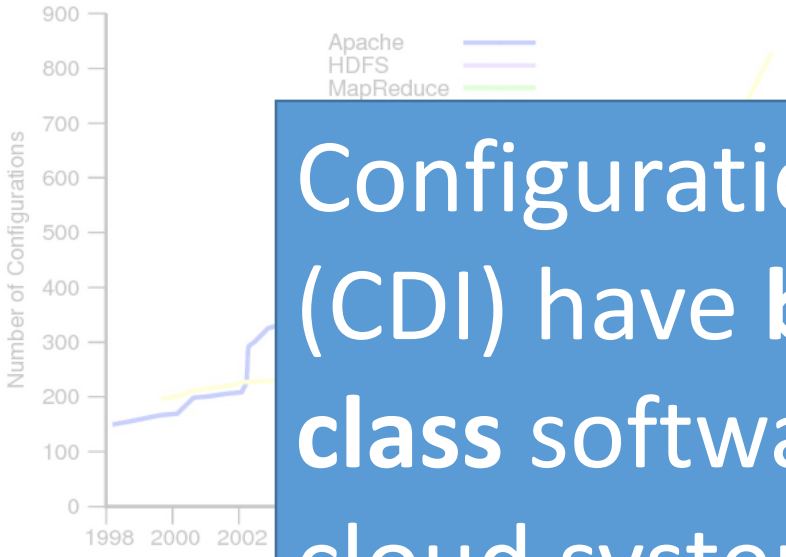
"is the parameter helpful?"
-Spark-25676

"can we reuse an existing parameter?"
-HDFS-13735

"what is a reasonable default value?"
-HBase-19148

Struggling to design and implement configurations

Motivation



"is the configuration helpful?"

Spark-25676

parameter?"

HDFS-13735

ult value?"

HBase-19148

Configuration Design and Implementation (CDI) have been largely overlooked as first-class software engineering endeavors in cloud systems

complexity increases rapidly over time

Previous: misconfiguration detecting and diagnosing

Struggling to design and implement configurations

Motivation

- Parameter: spark.sql.codegen.cache.maxEntries (default = 100)
- Evolution activity: Parameterization

```
Private val cache = CacheBuilder.newBuilder()
-     .maximumSize(100)
+     .maximumSize(SQLConf.get.codegenCacheMaxEntries)
```

- Rationale:

*"The cache 100 in CodeGenerator is **too small for realtime streaming calculation**, which is mostly more complex in one driver, and **performance** sensitive."*

Motivation

- Parameter: spark.sql.codegen.cache.maxEntries (default = 100)
- Evolution activity: Parameterization

Evolution history can help us:

- **Understand the rationale for the changes**
- **Learn design lessons and principles**

- Rationale:

*"The cache 100 in CodeGenerator is **too small for realtime streaming calculation**, which is mostly more complex in one driver, and **performance** sensitive."*

Contributions

- Study and Insights.
  - Insights that motivate future research on reducing misconfigurations.
- Taxonomy.
  - A taxonomy of cloud system configuration design and implementation evolution.
- Dataset and code.
  - https://github.com/xlab-uiuc/open-cevo

Methodology

- **4** large-scale, widely-used, actively-maintained open-source cloud systems
- **1178** configuration-evolved commits spanning **2.5** years

SUBJECT	#DESCRIPTION	#PARAMS	#ALLC	#STUDIEDC
HDFS	File system	560	1618	221
HBase	Database	218	3516	268
Spark	Data processing	442	6194	602
Cassandra	Database	220	1868	87

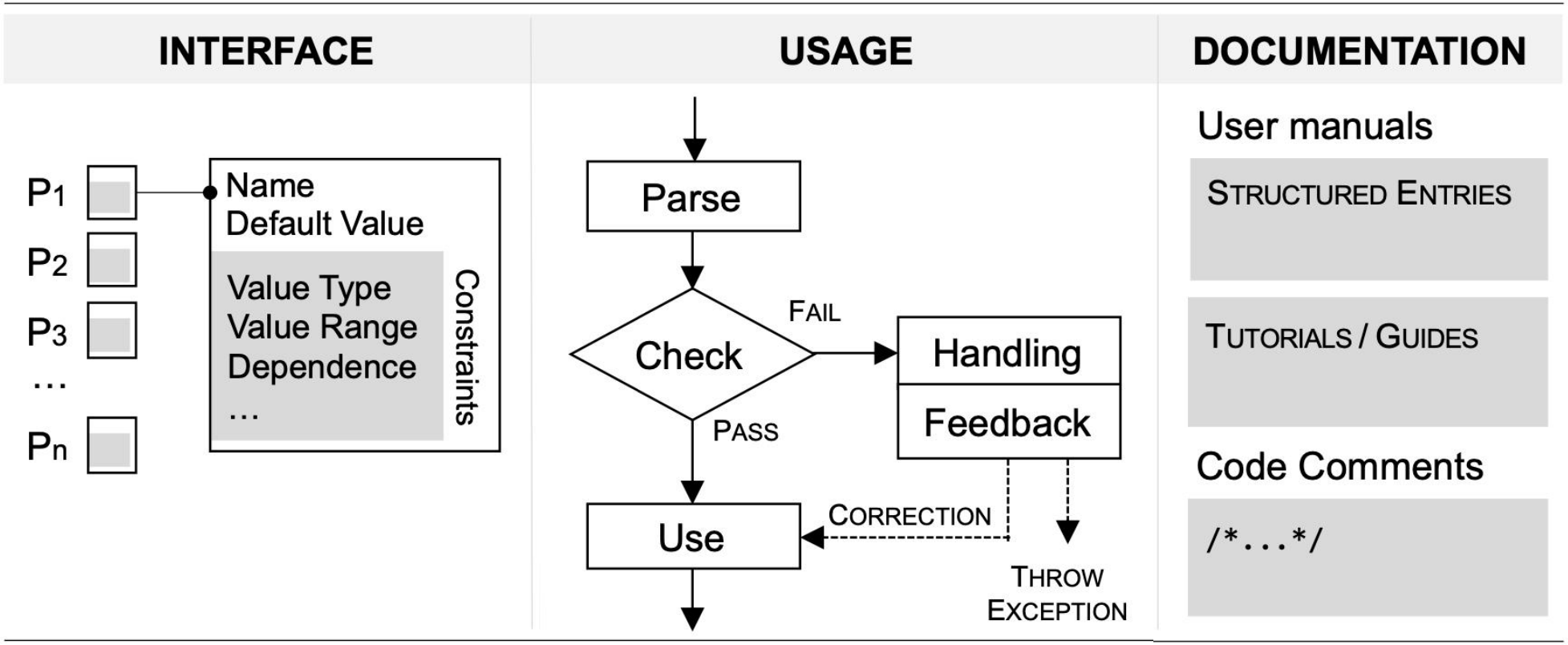
Methodology

- Code diff
  - Commit identification
  - Source code level categorization

- Issue discussion
  - Background
  - Rationale

Taxonomy

- 3 categories of CDI
- 16 evolution activities



Interface

- Over 50% of parameterizations were driven by severe consequences.
  - **Performance** tuning and **reliability** are common rationales.
  - Triggering **use cases** were often poorly discussed or documented.

- Only 28.1% of default-value changes mentioned systematic testing; 31.3% of default changes chose values that work around reported issues (without systematic assessment).

> *It probably makes sense to set it to something lower.* *-Spark-24297*
> *I'm thinking something like 3000 or 5000 would be safer.* *-HBase-18023*

Usage

- Over 50% of checks added as afterthoughts are basic (non-emptiness and value-range checks)

HBASE-18161

```
+  if (writeTable==null || writeTables.isEmpty()){
+     throw new IllegalArgumentException(
+         "Configuration parameter "  +
+         OUTPUT_TABLE_NAME_CONF_KEY + "cannot be empty")}
```

- Throwing exceptions is common for handling misconfigurations; auto-correction is not common, missing opportunities to help users handle errors.

Usage

- Enhance configuration-related log/exception messages by including related parameters and providing guidance.
  - We introduce 4 levels of message feedback quality

    L4: Contain parameter names and provide guidance for fixing
- Parameter reuse leads to various inconsistencies.

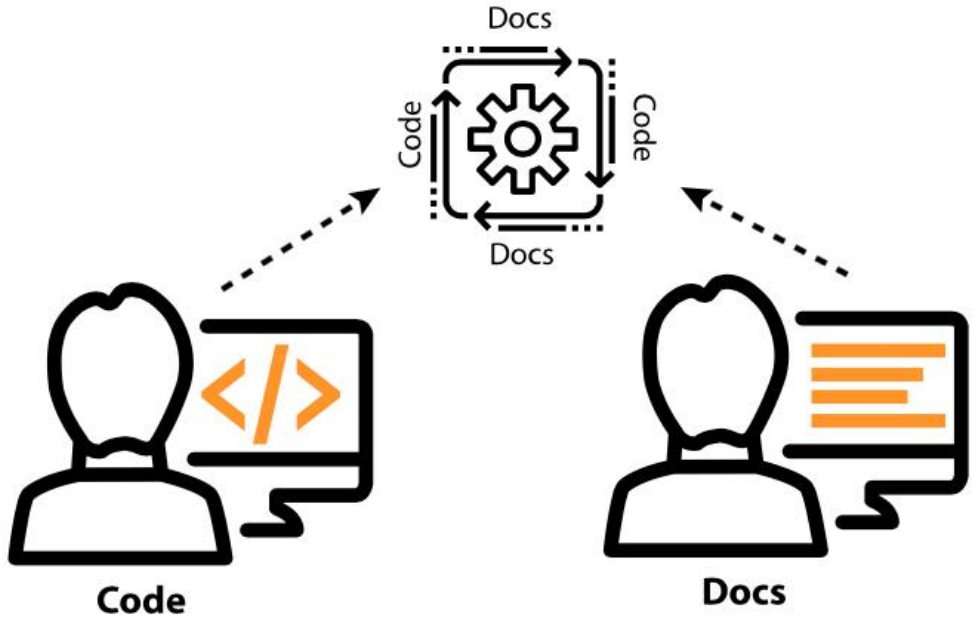
HBASE-24190

```
+    if (peerConf.get("hbase.security.authentication")
+          .equals("kerberos"))   {…}

     isSecurityEnabled = "kerberos".equalsIgnoreCase(
          conf.get(”hbase.security.authentication"));
     if   (isSecurityEnabled)  {…}
```

Documentation

- Configuration use cases, parameter constraints and dependencies between parameters are commonly added to documents.



Thanks! Q&A