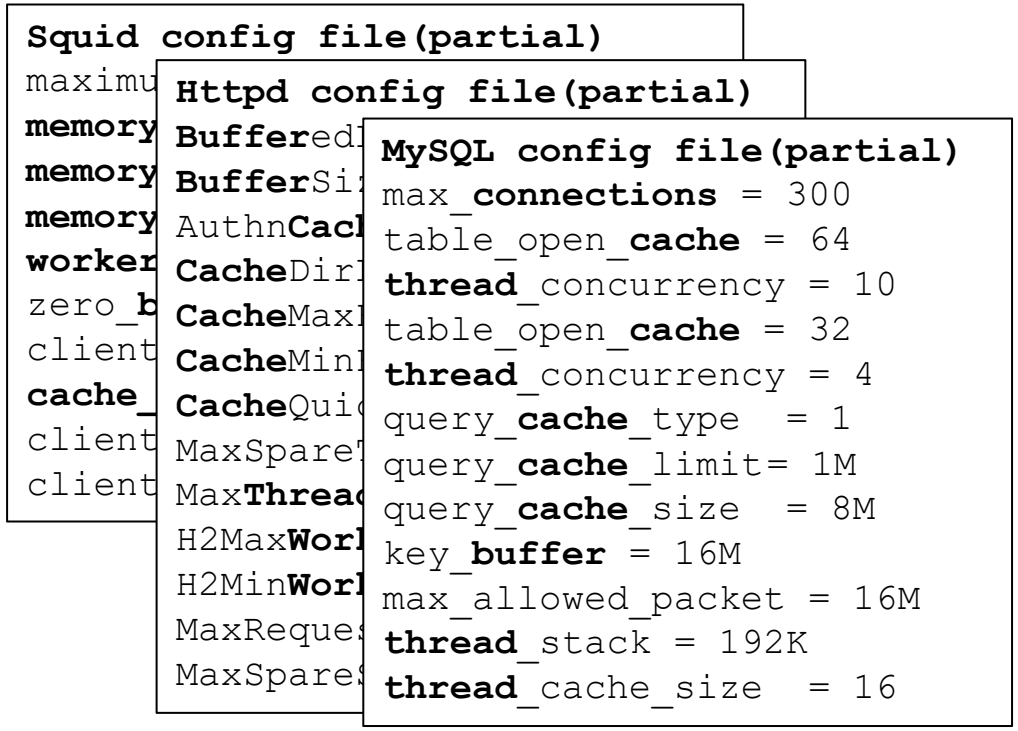
Who is in Charge here? Understanding How Runtime Configuration Affects Software along with Variables&Constants

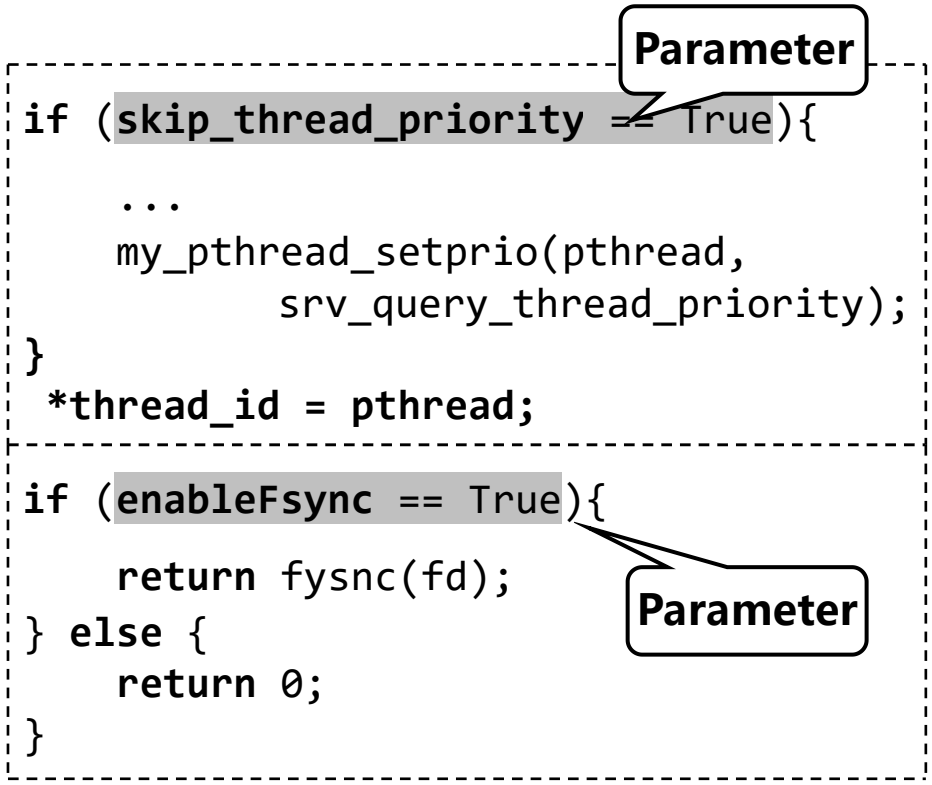
Chaopeng Luo, Yuanliang Zhang, Haochen He, Zhouyang Jia, Teng Wang, Shulin Zhou, Si Zheng, Shanshan Li

APSEC'24

Runtime configuration parameter

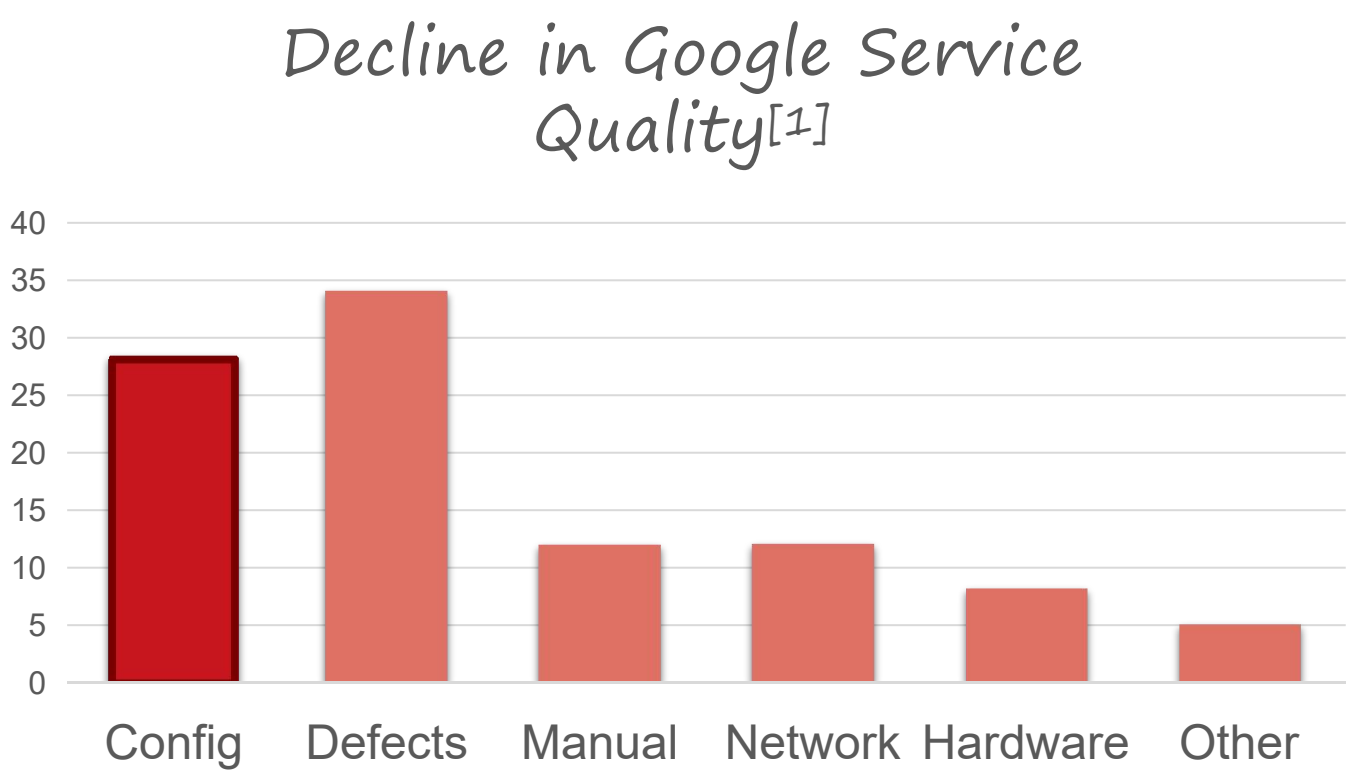


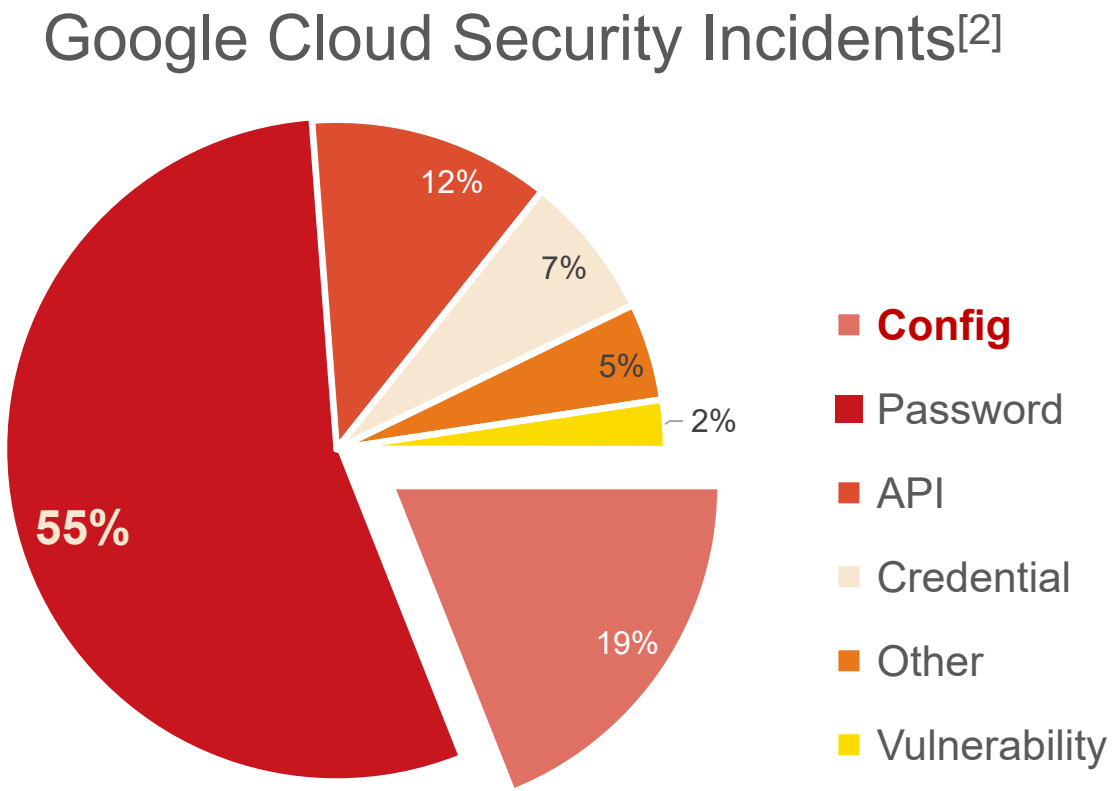
Configuration files



Configuration-related code

Configuration-related issues frequently occur





[1] Barroso et al. The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines, 3rd Edition [J]. **Synthesis Lectures on Computer Architecture. 2018.**

[2] August 2023 Threat Horizons Report Provides Cloud-Focused Cybersecurity Insights and Recommendations.

Misconfiguration prevention

Parameter

```
/* HBase/TableDescriptorChecker.java */
if (writePacketSize >
      PacketReceiver.MAX_PACKET_SIZE){
  LOG.warn(
    "write packet exceeds max byte
  ...
}
PackeChunkSize(writePacketSize, …);
...
```

Value check

Format check

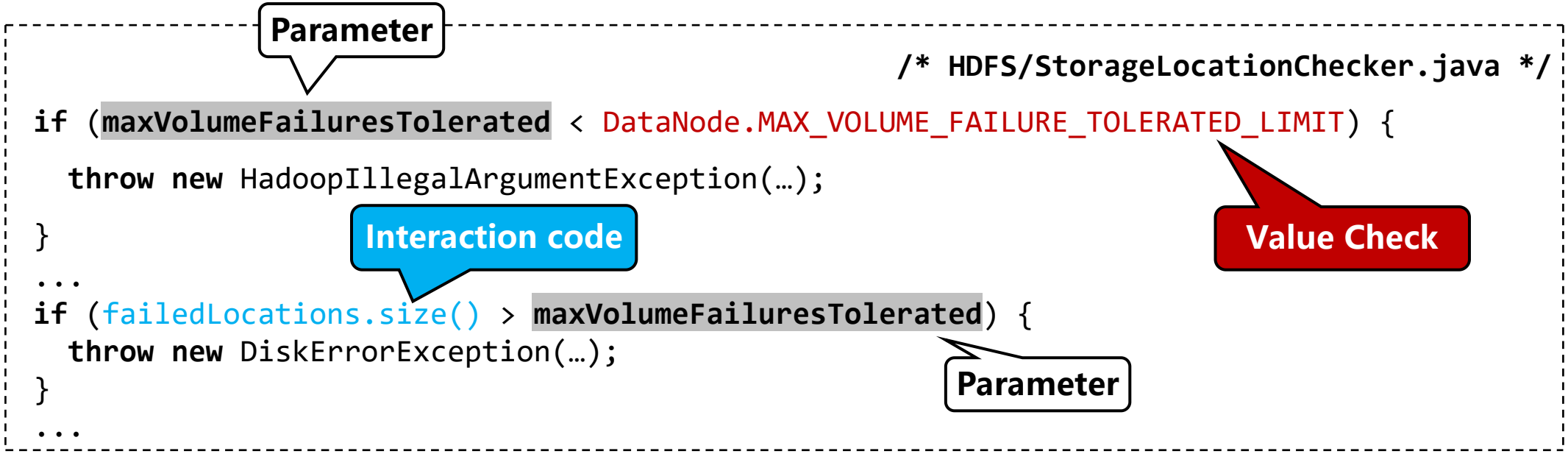
```
/* Httpd/mod_proxy.c */
if (ap_timeout_parameter_parse(arg,
      conf->async_idle_timeout, "s"){
    return "Timeout has wrong format";
}
idle_timeout = conf->async_idle_timeout;
...
```

Parameter

The developers added **correctness checks** for the configuration

Root cause of configuration-related issues

Interactions of Parameter & Constant & Variable (PCV Interaction)



Real-world example

HBASE-24544: Recommend upping zk jute.maxbuffer in all but minor installs

Configuration parameter: jute.maxbuffer = 1M (default value)

Workload: For recovery, there are hundreds of WALs to be read.

Sanity Check (parsing Stage)

```
protected void initProperties() throws IOException {
    try {
        packetLen = clientConfig.getInt(ZKConfig.JUTE_MAXBUFFER, …);
    } catch (Exception e) {
        LOG.error("… can not be parsed to int");
        throw new IOException();
    }
}
void readLength() throws IOException {
    int len = incomingBuffer.getInt();
    if (len < 0 || len > packetLen) {
        …
    }
    incomingBuffer = ByteBuffer.allocate(len);
}
```

java.io.IOException:
Packet is out of range!

Interaction causes severe consequence!

Consequences: power outage and service crashes down.

Contributions

- A study on **how configuration affects software** at runtime
  - 705 parameters collected from 10 software systems
- **Findings and insights** on PCV interactions.
  - Effects and potential problems
- An available **dataset** of PCV interactions.
  - https://github.com/PCVAnonymous/PCVStudy

Study methodology

- Study on PCV interaction from ten large-scale software systems
  - Rana

Software	Category	Lang.	# P	# Ps
Httpd	Web server	C	557	111
PostgreSQL	Database	C	251	50
Nginx	Proxy server	C	480	96
MySQL	Database	C++	390	78
HBase	Database	Java	174	35
Hive	Database	Java	484	97
HDFS	File system	Java	463	93
Yarn	Resource manager	Java	450	90
MapReduce	Data processing	Java	168	34
Zookeeper	Configuration manager	Java	154	21
Total	/	/	3523	705

Study methodology

- Trace the propagation of parameters
  - 851 interaction code snippets

Parsing

```
/* HBase/RpcServer.java */
long keyUpdateInterval =
  conf.getLong("hbase.auth.key.update.interval", 24 * 60 * 60 * 1000);
```

```
/* HBase/RpcServer.java */
protected AuthenticationTokenSecretManager createSecretManager() { …
  return new AuthenticationTokenSecretManager(…,keyUpdateInterval, …);
}
```

```
/* HBase/AuthenticationTokenSecretManager.java */
public AuthenticationTokenSecretManager(…) {
  …
  this.keyUpdateInterval = keyUpdateInterval;
}
```

Propagation

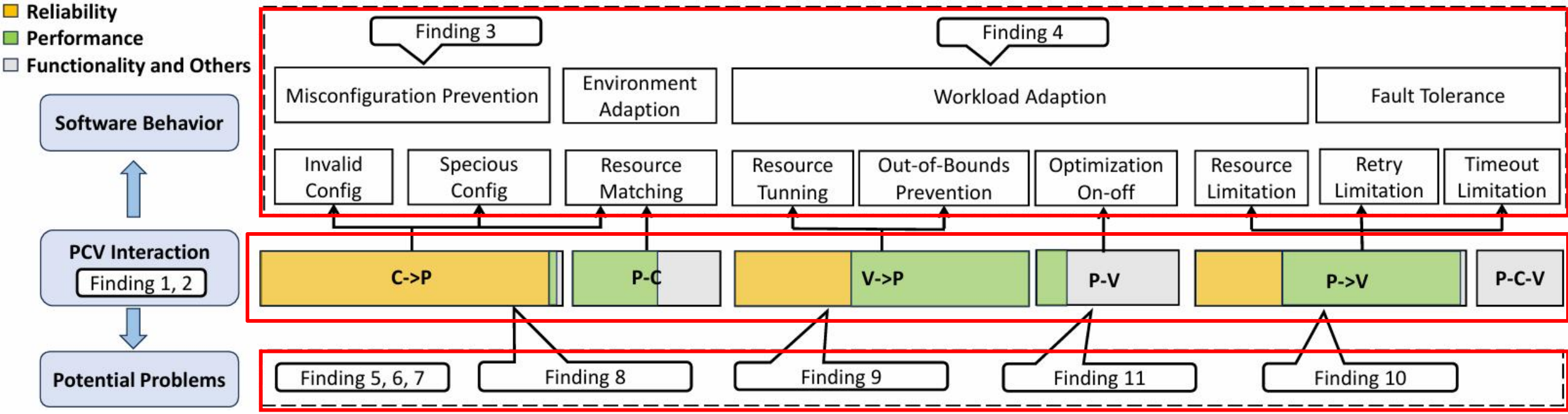
```
/* HBase/AuthenticationTokenSecretManager.java */
public void run() {
  …
  if (localLastKeyUpdate + keyUpdateInterval < now) {  … }
}
```

Usage

Research questions

- RQ1: Are PCV interactions common in software?
- RQ2: What are the **types and patterns** of PCV interactions?
- RQ3: How do PCV interactions **take effects** on software?
- RQ4: What are the **potential problems** behind PCV interactions?

Overall structure of our study



RQ1: Prevalence

- Finding 1: PCV interactions are **prevalent**
  - Only **33.6%** of parameters can **independently** affect software behaviors

```
/* HBase/RpcServer.java */

int opThreads = conf.getInt(…);
ThreadPoolExecutor pool =
  ProcedureMember.defaultPool(…, opThreads, …);
```

Parameter

Independently influence

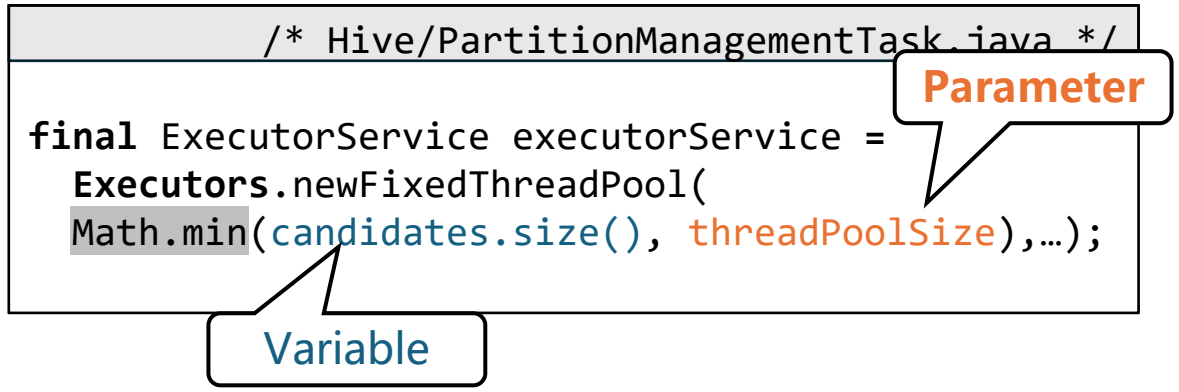
```
/* Hive/PartitionManagementTask.java */

final ExecutorService executorService =
  Executors.newFixedThreadPool(
  Math.min(candidates.size(), threadPoolSize),…);
```

Parameter

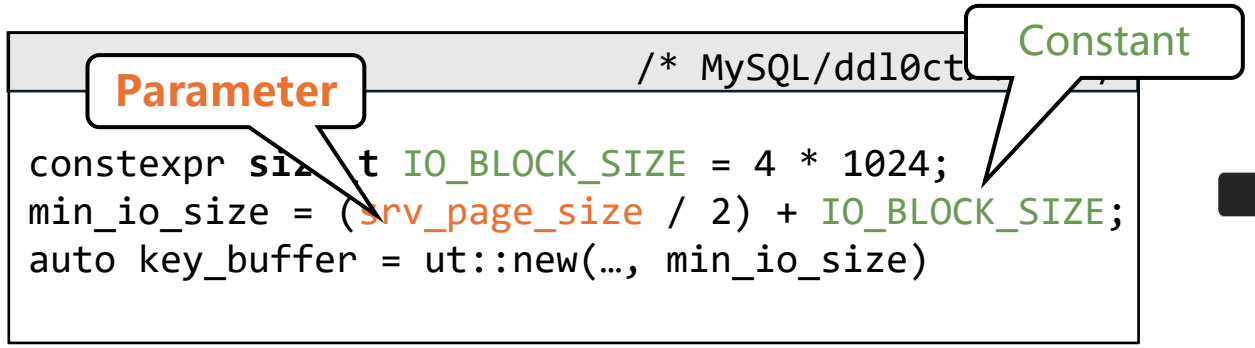
Jointly influence

RQ2: Types and patterns

- **Seven** types of interaction code snippets



P is constrained by V



P, C take equivalent place

RQ3: Effect

- Finding 2: great impact on software **performance** and **reliability**
  - Only **a small portion (<20%)** is dedicated to implementing the logical functionalities
- Finding 3: Efforts are primarily for **misconfiguration prevention**
  - Also for environment adaption

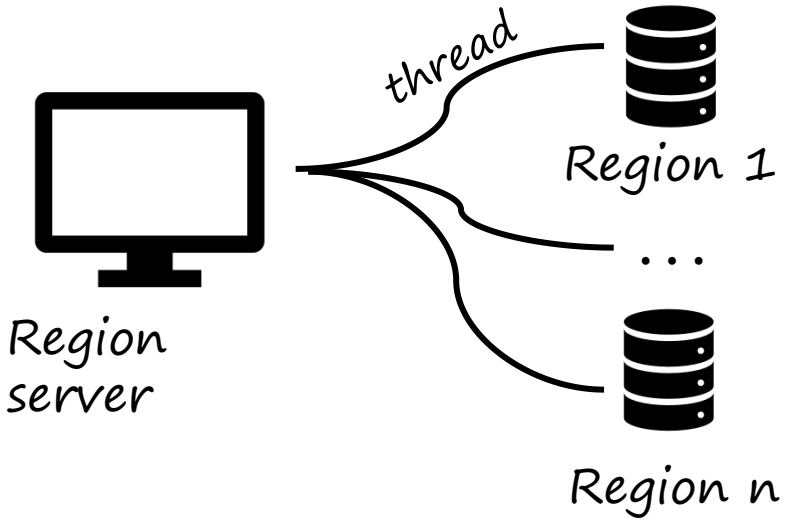
```
/* MySQL/ut0dbg.cc */

constexpr size_t IO_BLOCK_SIZE = 4 * 1024;
min_io_size = (srv_page_size / 2) + IO_BLOCK_SIZE;
auto key_buffer = ut::new(…, min_io_size)
```

Matching the I/O block size

RQ3: Effect

- Finding 4: P&V interactions are used for **workload adaption**
  - About 80% are to prevent excessive resource utilization

```
1  int regionNumber = newRegions.length;
2  int maxThreads = Math.min(regionNumber,
3        conf.getInt("hbase.hregion.open.and.init.threads.max", 16));
4  Threads.getBoundedCachedThreadPool(maxThreads,……);
```



Resource conservation

RQ4: Potential problems of PCV interactions

- Finding 5: prone to **bad consequence (56.5%)**
  - Runtime error, performance degradation and unexpected results.
- Finding 6: lack of **log** information
  - Only **15.3% of cases** explicitly inform users about the interactions.
- Finding 7: lack of **on-the-fly update** support
  - Only 7 of 240 parameters in Java software support updates at runtime.

Potential problems of P&C

- Finding 8: the parameters may be **overwritten** by the constants

```
-       /* Force a truncate of the history list. */
-       n_pages_purged = trx_purge(1, srv_purge_batch_size, true);
+       /* This trx_purge is called to remove any undo records (added by
+       back
+       srv_
+       dela
+       (whi
+       remo
+       cons
```

```
@@ -203,6 +203,9 @@ pre_init_event_thread(THD* thd)
203         thd->version= refresh_version;
204         thd->set_time();
205
206 +       /* Do not use user-supplied timeout value for system threads. */
207 +       thd->variables.lock_wait_timeout= LONG_TIMEOUT;
208 +
```

Potential problems of P&V

- Finding 9: **Not adapting** the workload
  - Good practice: parameter adjustment based on historical data

```
                                                          /* MySQL#31965404 */
for (size_t i = 0; i < undo::spaces->size(); ++i) {
    ...
-   const auto n_pages = SRV_UNDO_TABLESPACE_SIZE_IN_PAGES;// Use constant(C)
+   auto n_pages = UNDO_INITIAL_SIZE_IN_PAGES;  // Use param(P) in normal execution
+   auto space = fil_space_get(old_space_id);   // Get historical workload status(V)
+   if (space->m_undo_extend > UNDO_INITIAL_SIZE_IN_PAGES &&
+     space->m_last_extended.elapsed() < 1000) {
+     n_pages = fil_space_get_size(old_space_id) / 4;
+   }
    fil_ibd_create(…, n_pages);
}
```

Calculated from historical workload value

UNDO tablespace extends aggressively

Potential problems of P&V

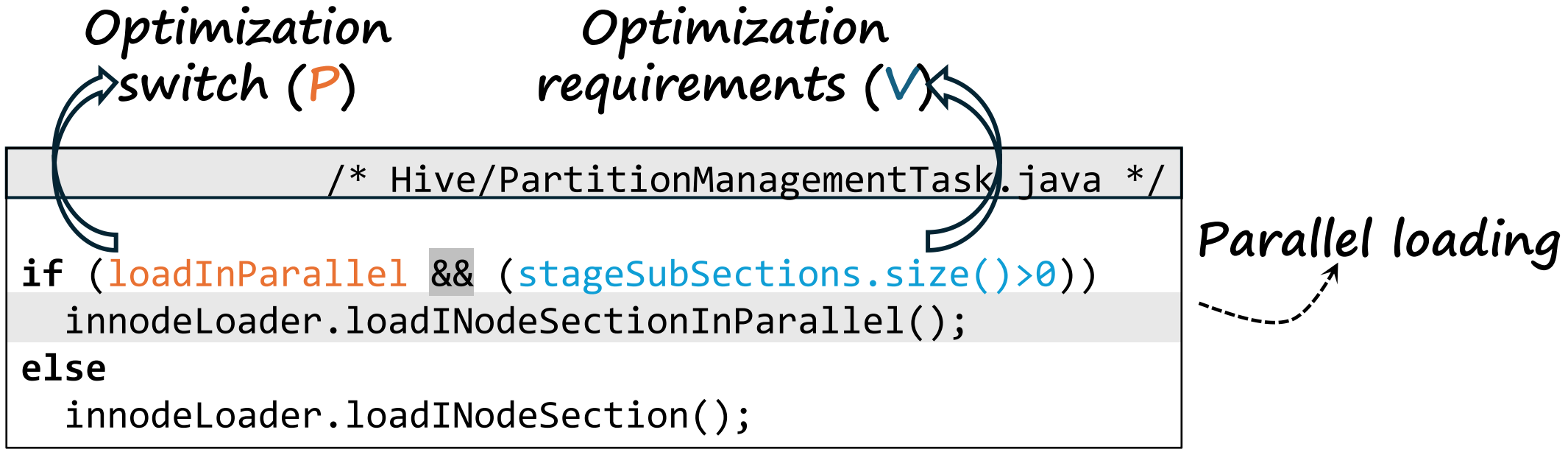
- Finding 10: **Inappropriate threshold** in P → V
  - Good practice: elastic threshold

```
                                                    /* Yarn/FederationClientInterceptor.java */
int initSize = Math.min(INITIAL_THREAD_POOL_SIZE, maxThreadPoolSize);
threadPool = new ThreadPoolExecutor(initSize, …);          // Initialize with a small constant(C)
while (…) {
  int threadPoolSize = threadPool.getCorePoolSize();
  int nodeNum = allNodes.size();                                  // Get workload status (V)
  int idealThreadPoolSize = Math.min(maxThreadPoolSize, nodeNum);   // Combine P and V
  if (threadPoolSize < idealThreadPoolSize)
        ··· Extend thread pool based on P and V
}
```

Not enough for runtime workload

Potential problems of P&V

- Finding 11: **Missing optimization opportunities** in P-V
  - Developers tend to tune the option to **off** conservatively



66.7% default values of optimization parameters are false

Conclusion

- New perspective of how configuration affects software at runtime
- Our study reveals interesting findings
  - "Double-edge" of PCV interactions
  - Good practice from developers
- Available dataset for future works

https://github.com/PCVAnonymous/PCVStudy