



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

```
Squid config file(partial)
maximum
memory
memory
memory
worker
zero_b
client
cache_
client
client

Httpd config file(partial)
Buffered
BufferSiz
AuthnCac
CacheDir
CacheMax
CacheMin
CacheQuic
MaxSpare
MaxThread
H2MaxWorl
H2MinWorl
MaxReques
MaxSpare

MySQL config file(partial)
max_connections = 300
table_open_cache = 64
thread_concurrency = 10
table_open_cache = 32
thread_concurrency = 4
query_cache_type = 1
query_cache_limit= 1M
query_cache_size = 8M
key_buffer = 16M
max_allowed_packet = 16M
thread_stack = 192K
thread_cache_size = 16
```

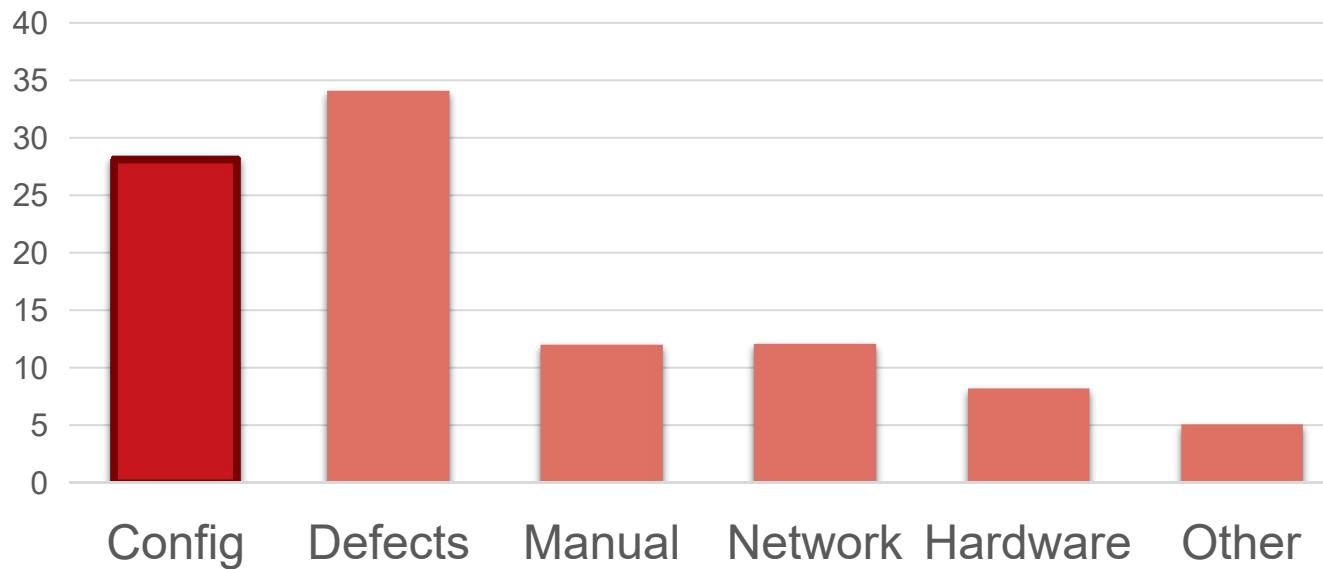
Configuration files

```
if (skip_thread_priority == True){  
    ...  
    my_pthread_setprio(pthread,  
                        srv_query_thread_priority);  
}  
*thread_id = pthread;  
  
if (enableFsync == True){  
    return fysnc(fd);  
} else {  
    return 0;  
}
```

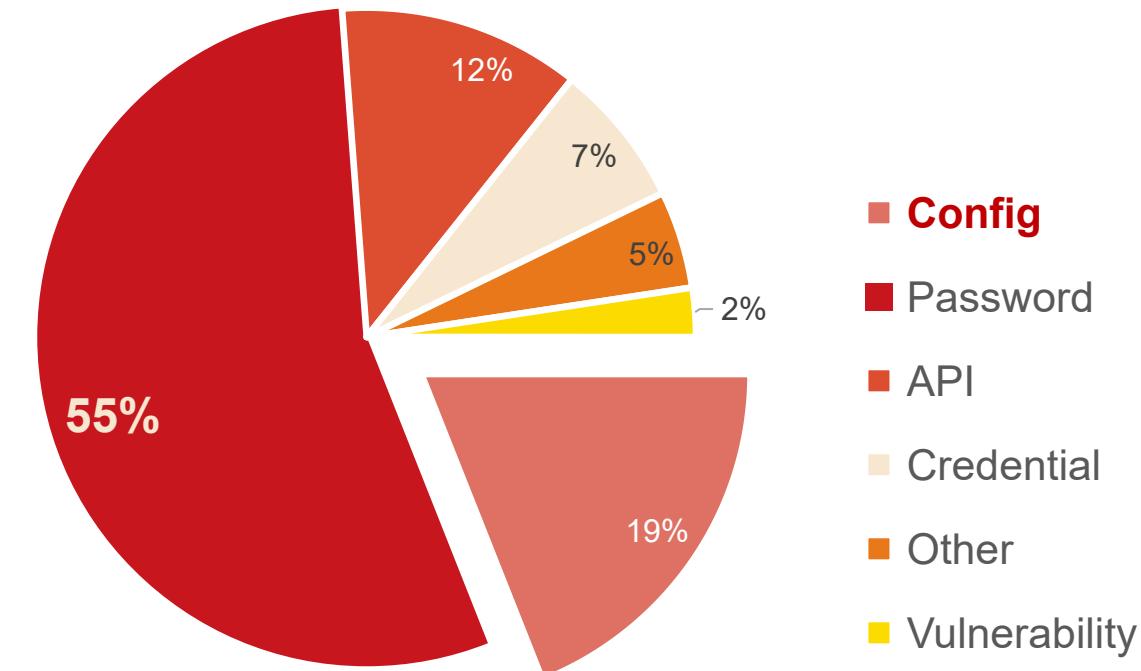
Configuration-related code

Configuration-related issues frequently occur

Decline in Google Service Quality^[1]



Google Cloud Security Incidents^[2]



[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

The diagram illustrates two code snippets with annotations:

- Left Snippet (HBase/TableDescriptorChecker.java):**
 - A red box labeled "Parameter" points to the variable `writePacketSize`.
 - A red box labeled "Value check" points to the condition `if (writePacketSize > PacketReceiver.MAX_PACKET_SIZE){}`.
 - A red box labeled "Format check" points to the parameter `conf->async_idle_timeout, "s"` in the right snippet.
- Right Snippet (Httpd/mod_proxy.c):**
 - A red box labeled "Parameter" points to the parameter `conf->async_idle_timeout, "s"`.
 - A red box labeled "Format check" points to the annotation above the parameter `/* Timeout has wrong format */`.

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

```
/* 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;
...
```

The developers added correctness checks for the configuration

Root cause of configuration-related issues

Interactions of Parameter & Constant & Variable (PCV Interaction)

The diagram illustrates the interactions between parameters, constants, and variables in the `HDFS/StorageLocationChecker.java` code. A dashed-line box encloses the code. Inside, a callout labeled "Parameter" points to the variable `maxVolumeFailuresTolerated`. Another callout labeled "Interaction code" points to the conditional logic. A red arrow labeled "Value Check" points from the constant `DataNode.MAX_VOLUME_FAILURE_TOLERATED_LIMIT` to the comparison in the first if-statement. A final callout labeled "Parameter" points to the variable `failedLocations.size()` used in the second if-statement.

```
/* HDFS/StorageLocationChecker.java */
if (maxVolumeFailuresTolerated < DataNode.MAX_VOLUME_FAILURE_TOLERATED_LIMIT) {
    throw new HadoopIllegalArgumentException(...);
}
...
if (failedLocations.size() > maxVolumeFailuresTolerated) {
    throw new DiskErrorException(...);
}
...
```

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);
}
```

Interaction causes
severe consequence!

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

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

- Rand

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

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

/* HBase/RpcServer.java */

```
protected AuthenticationTokenSecretManager createSecretManager() { ...  
    return new AuthenticationTokenSecretManager(..., keyUpdateInterval, ...);  
}
```

/* HBase/RpcServer.java */

```
public AuthenticationTokenSecretManager(...) {  
    ...  
    this.keyUpdateInterval = keyUpdateInterval;  
}
```

Propagation

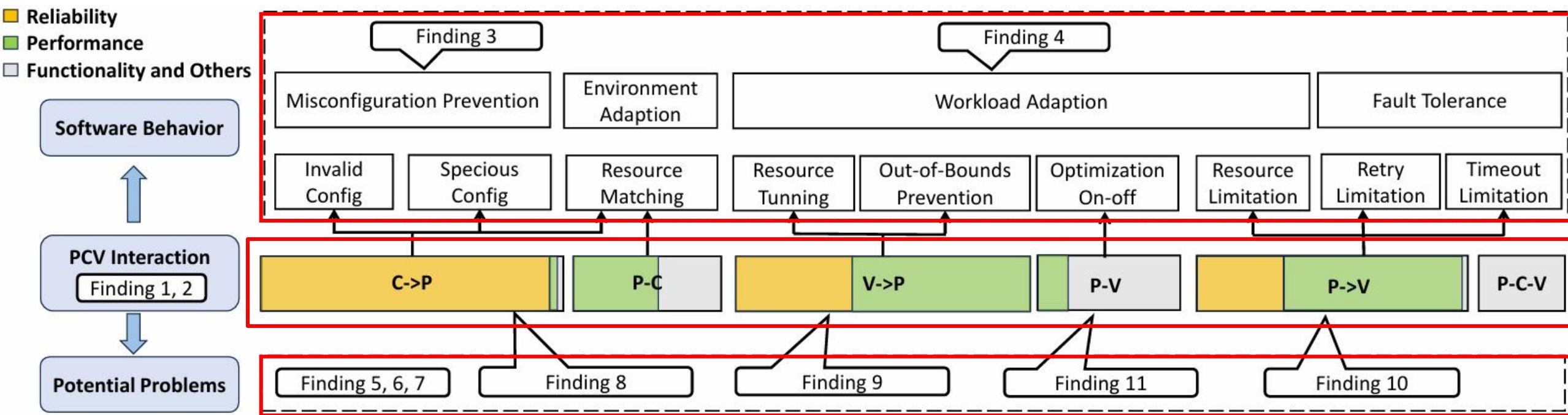
```
public void run() {  
    ...  
    if (localLastKeyUpdate + keyUpdateInterval < now) { ... }  
}
```

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, ...);
```

Independently influence

Parameter

```
/* Hive/PartitionManagementTask.java */  
  
final ExecutorService executorService =  
    Executors.newFixedThreadPool(  
        Math.min(candidates.size(), threadPoolSize), ...);
```

Jointly influence

Parameter

RQ2: Types and patterns

□ Seven types of interaction code snippets

```
/* Hive/PartitionManagementTask.java */  
  
final ExecutorService executorService =  
    Executors.newFixedThreadPool(  
        Math.min(candidates.size(), threadPoolSize), ...);
```

Variable → Parameter



P is constrained by V

```
/* MySQL/ddl0ct */  
  
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)
```

Parameter → Constant



P, C take equivalent place

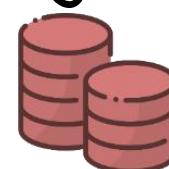
RQ3: Effect

- Finding 2: great impact on software performance and reliability
 - Only a small portion (<20%) is dedicated to implementing the
- Filaginab fznC R&Gbitiexe primarily for misconfiguration prevention

● Also for environment adaptation */

```
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)
```

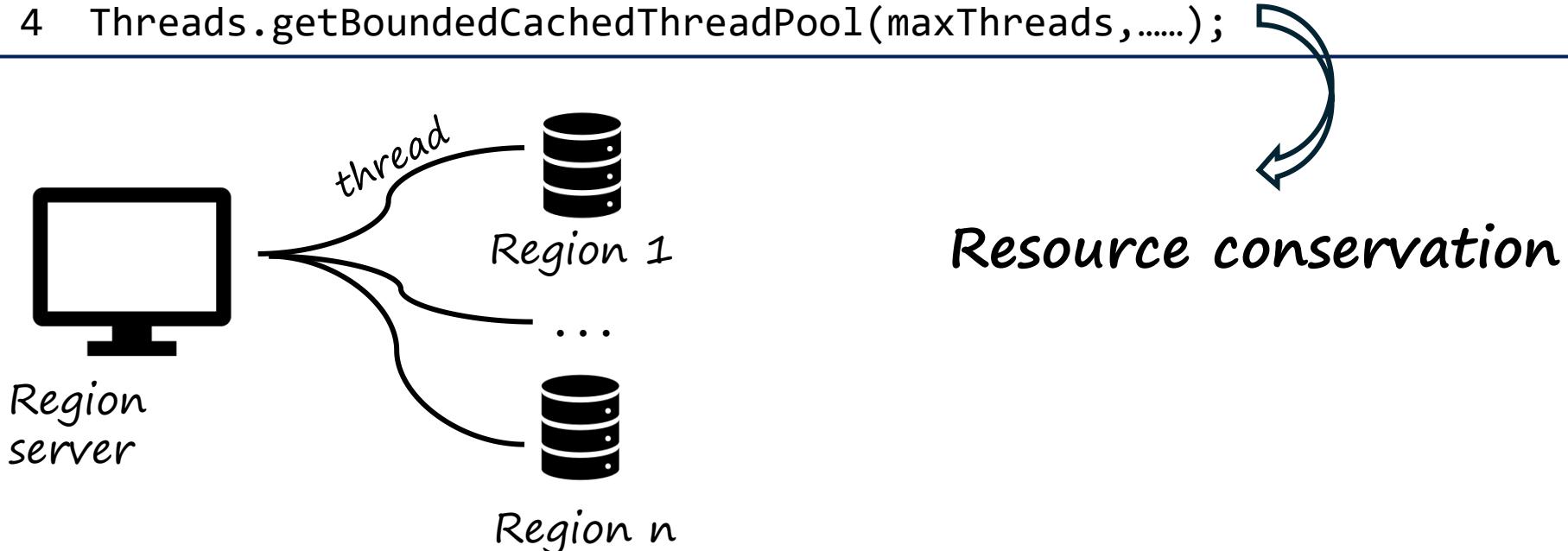
Matching the I/O block size



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,.....);
```



## 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... @@ -203,6 +203,9 @@ pre_init_event_thread(THD* thd)
+ srv_
+ delay 203 thd->version= refresh_version;
+ (whi 204 thd->set_time();
+ remo 205
+ cons 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  $P \rightarrow V$ 
  - 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

The diagram shows a code snippet from `Hive/PartitionManagementTask.java`. The code contains an `if` statement that checks if `loadInParallel` is true and if there are sub-sections to load in parallel. If both conditions are met, it calls `innodeLoader.loadINodeSectionInParallel()`; otherwise, it calls `innodeLoader.loadINodeSection()`.

Annotations on the code:

- A red arrow points from the text "Optimization switch (P)" to the `loadInParallel` variable.
- A red arrow points from the text "Optimization requirements (VX)" to the `stageSubSections.size() > 0` condition.
- A dashed red arrow points from the text "Parallel loading" to the `loadINodeSectionInParallel()` method call.

```
/* Hive/PartitionManagementTask.java */
if (loadInParallel && (stageSubSections.size()>0))
 innodeLoader.loadINodeSectionInParallel();
else
 innodeLoader.loadINodeSection();
```

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/PCVS  
study](https://github.com/PCVAnonymous/PCVSstudy)