An Empirical Study on Configuration Errors in Commercial and Open Source Systems

Zuoning Yin, Xiao ma, Jing Zheng, Yuanyuan Zhou
University of California At San Diego

Lakshmi N. Bairavasundaram, Shankar Pasupathy
Netapp Inc.
Configuring computers is not easy
Configuring server systems is much harder
Root Causes of Customer Reported Issues

- Configuration: 31%
- Hardware Failure: 20%
- Customer Environment: 25%
- Bug: 15%
- User Knowledge: 9%
Who should take responsibility for configuration errors, users or developers?
• Developers do not think carefully when they design configuration interface

• Once an issues turns out to be a configuration error, developers move on
How to Reduce Configuration Errors?

- What kind of configuration errors do users make?
- Which types of configuration parameters are more error-prone?
- Which user actions may cause configuration errors?
- ......

We need to understand real-world configuration errors first
Objectives and Challenges

- Objectives
  - Understand the characteristics of real-world configuration errors
  - Reveal their implications to developers

- Challenges
  - Configuration errors are not recorded rigorously
  - Configuration errors are difficult to understand
Methodology

• Random sampling configuration errors
  • Choose resolved cases in recent 2 years
  • Ensure the sample size is big enough
  • Calculate statistical error
• Categorizing errors with best effort
  • Cross-validation among co-authors
  • Help from developers
## Data Source

<table>
<thead>
<tr>
<th>System</th>
<th>Number of Sampled Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Commercial</strong></td>
<td></td>
</tr>
<tr>
<td>COMP-A</td>
<td>309</td>
</tr>
<tr>
<td><strong>Open Source</strong></td>
<td></td>
</tr>
<tr>
<td>CentOS</td>
<td>60</td>
</tr>
<tr>
<td>MySQL</td>
<td>55</td>
</tr>
<tr>
<td>Apache</td>
<td>60</td>
</tr>
<tr>
<td>Open LDAP</td>
<td>62</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>546</strong></td>
</tr>
</tbody>
</table>
Limitation

• We study only user-reported errors
  • Configuration errors may be resolved in other means

• We focus on server-side systems
  • Other types of systems may have different characteristics
Software Configuration Errors

- Parameter Errors
- Compatibility Errors
- Other Errors
Parameter errors dominate
- Systems should expose as few configuration parameters as possible
- Automatic configuration is preferred

Parameter errors dominate
Other types of configuration errors are not negligible.
Parameter Errors

COMP-A
InitiatorName: iqn_DEV_domain
Lower-case only value
Error!

MySQL
AutoCommit = True
“True” value may affect performance
Parameter Errors

Illegal Parameters

InitiatorName: iqn_DEV_domain

- Lower-case only value
- Error!

Legal Parameters

MySQL

AutoCommit = True

- “True” value may affect performance
Illegal and legal parameters have similar contribution to parameter errors
• Illegal parameters are relatively easy to detect

• About half of parameter errors involve illegal parameters ("good" news!)
Illegal Parameters

Value Errors

Format Errors

Inconsistent Values Errors

Other Value Errors
Inconsistent values dominate illegal parameter errors for most systems.
All systems have format errors, in particular, Open LDAP has 69%
Inconsistent Value Errors

**PHP + MySQL**

- **PHP configuration:**
  - `mysql.max_persistent = 400`

- **MySQL configuration:**
  - `max_connections = 300`

- The value in PHP should not be bigger than the value in MySQL.

---

**MySQL**

- `log_output="Table"`
- `log=query.log`

Not consistent. They should be:

- `log_output="Table"`
  - or
- `log_output="File"
  - log=query.log`
• Value consistency constraints are error-prone; they account for most illegal parameter errors

• Consistency constraints could be across multiple systems, which is more difficult for users to follow
Format Errors

COMP-A

InitiatorName: iqn_DEV_domain

Lower-case only value Error

OpenLDAP

include-schema/ppolicy.schema
......
overlay ppolicy Missing

Apache

extension = mysql.so
......
extension = recode.so

recode.so must be put before mysql.so
• Format constraints are difficult to follow, especially non-intuitive ones, e.g., upper-case vs. lower-case or ordering

• Format errors are relatively easier to detect compared to value errors
Type

Reaction

Impact

Cause
Good Reaction #1

- **Symptom:** the user cannot create new directories in “/vol/vol1/data/”
- **Reaction:** the system prints this message:

```
[COMP-A - dir.size.max:warning]:
Directory /vol/vol1/data/ reached the maxdirsize Limit. Reduce the number of files or use the vol options command to increase this limit.
```
Good Reaction #2

MySQL

log_output="Table"
...
log=query.log

Patch:

+if (opt_logname
+ && !(log_output_options & LOG_FILE)
+ && !(log_output_options & LOG_NONE))
+ sql_print_warning("Although a path was specified
+ for the --log option, log tables are used. To enable
+ logging to files use the --log-output option.");
**Good Reaction #2**

Patch:

```c
+ if (opt_logname
+   && !(log_output_options & LOG_FILE)
+   && !(log_output_options & LOG_NONE))
+   sql_print_warning("Although a path was specified
+   for the --log option, log tables are used. To enable
+   logging to files use the --log-output option.");
```
Today’s systems do not react to configuration errors in a user-friendly way
Big portion of quiet failures makes diagnosis difficult
• Systems should avoid “bug-like” symptoms when configuration errors happen, such as quite failures, crash or hang

Big portion of quiet failures makes diagnosis difficult
Do systems react better to errors with “illegal” parameters?
Illegal parameter errors are handled better, but not good enough
How does message quality affect diagnosis time?
Message Quality

- Explicit Message
- Ambiguous Message
- No Message
Explicit messages significantly reduce diagnosis time
Messages are harmful if they are misleading
When does a configuration error happen?

- First-time Use
- Used-to-work
Complex systems are more likely to have configuration errors in the middle of lifetime
How do configuration errors affect system availability?

- Partially Unavailable
- Fully Unavailable
- Performance Degradation
Configuration errors can cause system full unavailability and performance degradation.
Performance configuration is especially difficult
• Performance parameters are more difficult to understand and set

• Diagnosing performance configuration issues are troublesome

Performance configuration is especially difficult
Other Characteristics

• Location and domain of parameter errors
• Number of involved/fixed parameters
• Complete categorization of illegal parameters and their distribution
• Complete analysis of causes of errors
• Details about compatibility errors and component errors
• Analysis across multiple directions
• More examples
**Related Work**

- **Prevention** (SmartFrog, Kardo, etc.)
- **Detection** (PeerPressure, Strider, etc.)
- **Diagnosis** (Chronus, AutoBash, ConfAid, etc.)
- **Tolerance** (AutoBash, Undo, etc.)
- **Validation** (Barricade, etc.)
- **Injection/testing** (ConfErr, etc.)
Summary

• Think from users’ point of view
• Users do not know the code
• Keep things simple
  • Expose fewer “knobs”
  • Use intuitive and simple rules
• Validate configuration proactively
• React decently when configuration errors happen
• Provide good feedback to users

• Record configuration errors

• Learn from mistakes
Thank you!