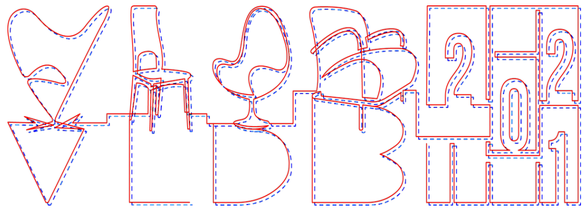


AutoGR: Automated Geo-Replication with Fast System Performance and Preserved Application Semantics

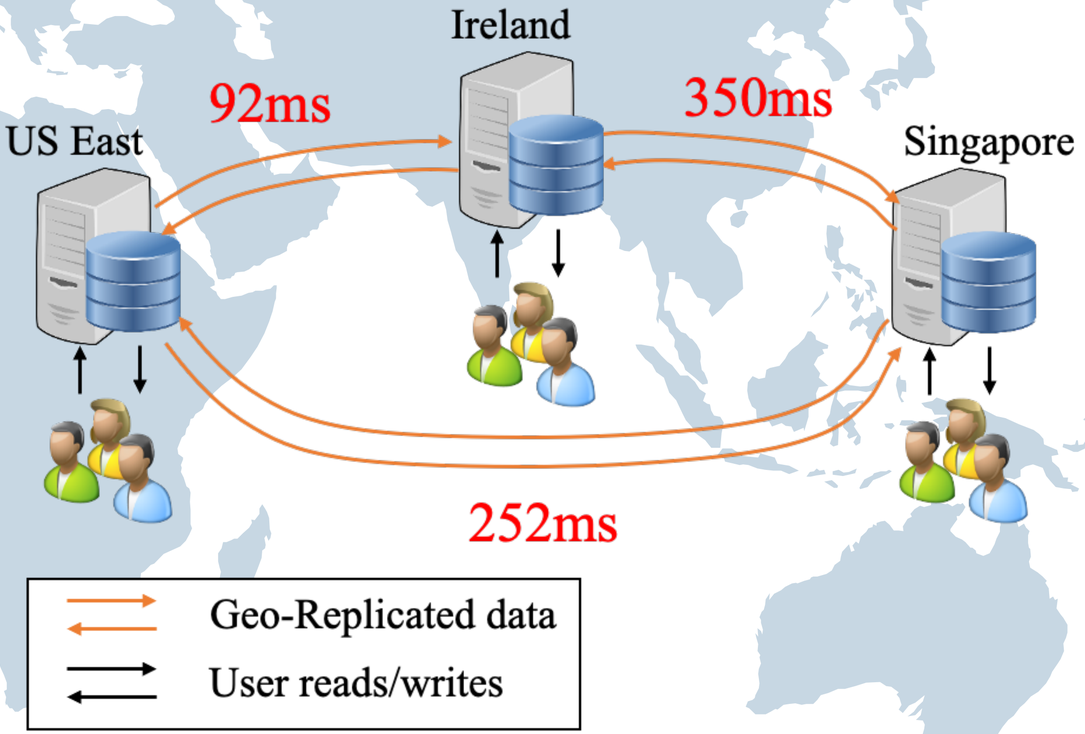
Jiawei Wang[1], Cheng Li[1], Kai Ma[1], Jingze Huo[1], Feng Yan[2], Xinyu Feng[3], Yinlong Xu[1]

Background


淘宝网
Taobao.com
facebook
Google
Ctrip
携程
Baidu 百度
WeChat
amazon
...

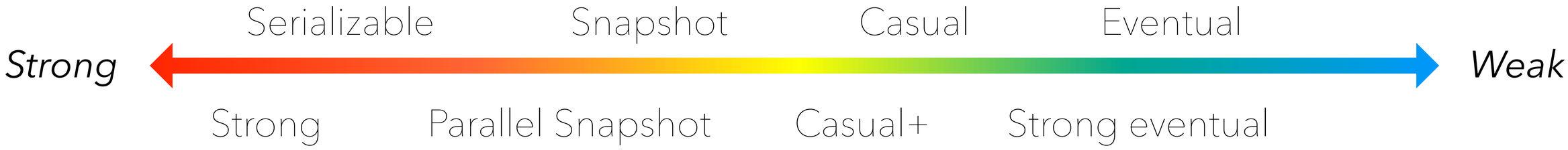
Motivation

- Geo-Replication as a major solution to cope with ever-growing user base for Internet services.

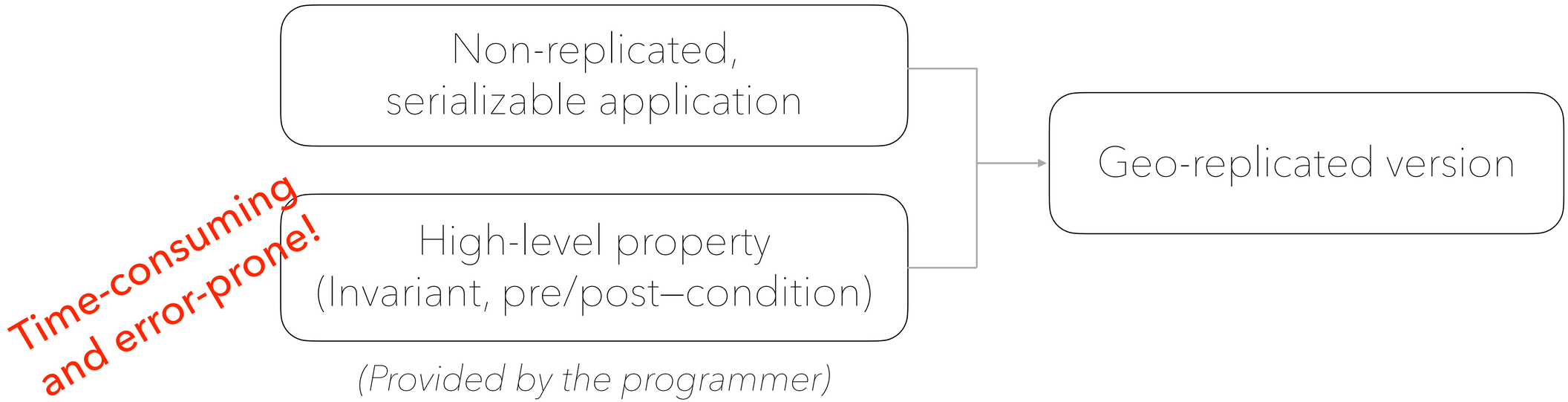


Strong consistency introduces high delay to user responses.

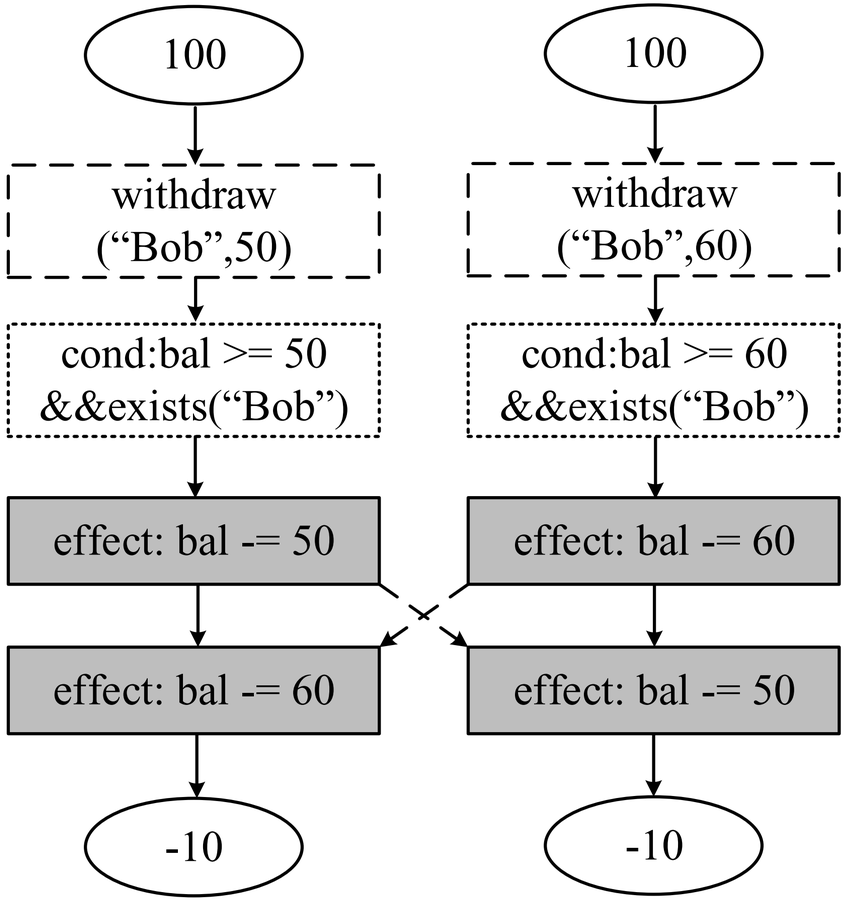
Related Work



Restriction-based fine-grained consistency model



Observation




```
void withdraw(Connection conn, String custName,
                        double amount) throws Exception {
    PreparedStatement stmt = conn.prepareStatement(
                        "SELECT * FROM ACCOUNTS WHERE name = ?");
    stmt.setString(1, custName);
    ResultSet rs = stmt.executeQuery();
    if (rs.next() == false) throw new Exception(“Invalid account”);
    long custId = rs.getLong(1);
    stmt = conn.prepareStatement("
            SELECT bal FROM SAVINGS WHERE custid = ?");
    stmt.setLong(1, custId);
    rs = stmt.executeQuery();
    if (rs.next() == false) throw new Exception(“No saving account”);
    double balance = rs.getDouble(1) – amount;
    if (balance < 0) throw new Exception(“Negative balance”);
    stmt = conn.prepareStatement(
                        "UPDATE SAVINGS SET bal =? WHERE custid =?");
    stmt.setDouble(1, balance); stmt.setLong(2, custId);
    stmt.executeUpdate();
    conn.commit();
}
```

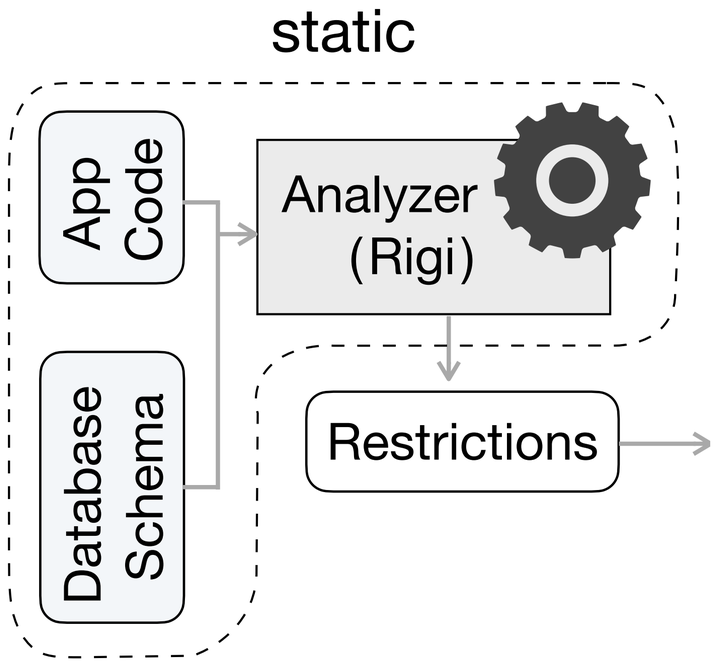
Application-specific invariants are already implicitly reflected in the programs.

AutoGR

static

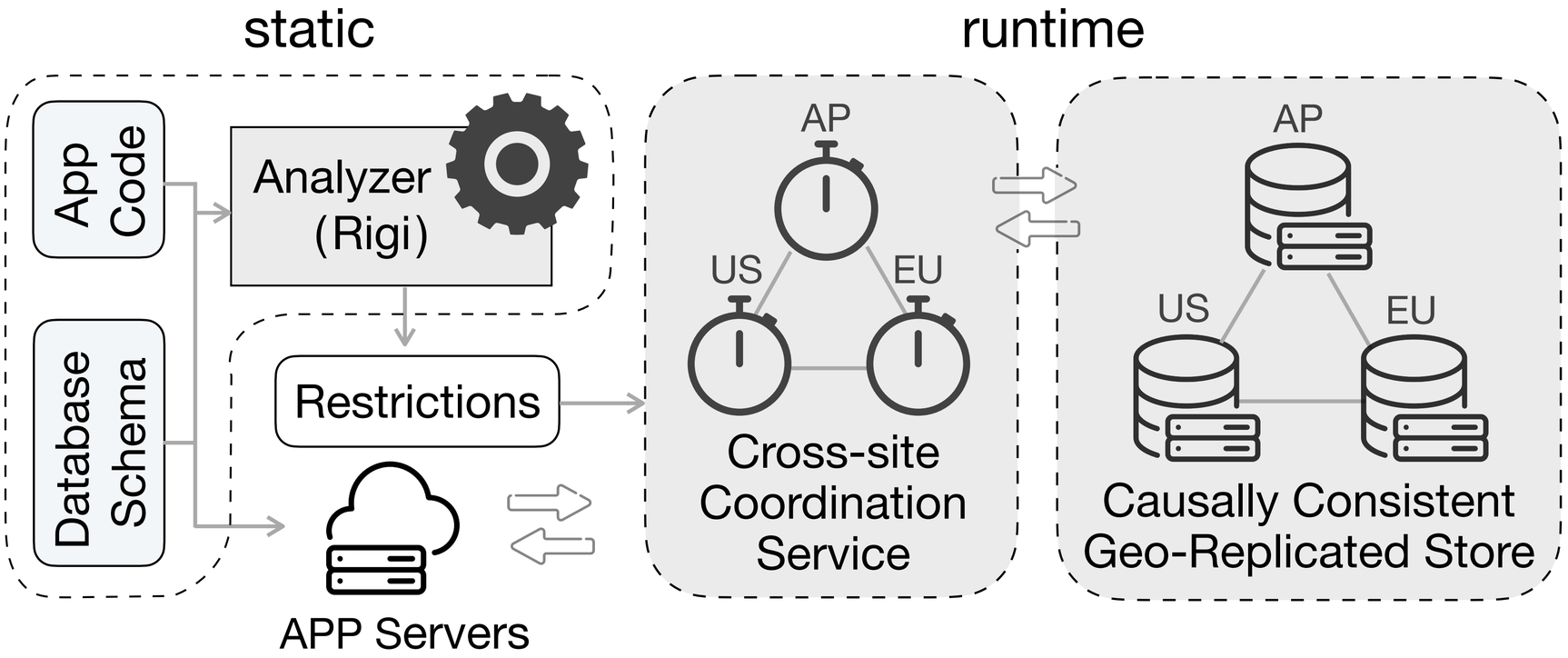
runtime

AutoGR



The static analyzer Rigi identifies a minimal set of ordering restrictions that must be ensured so that the intended semantics are not violated.

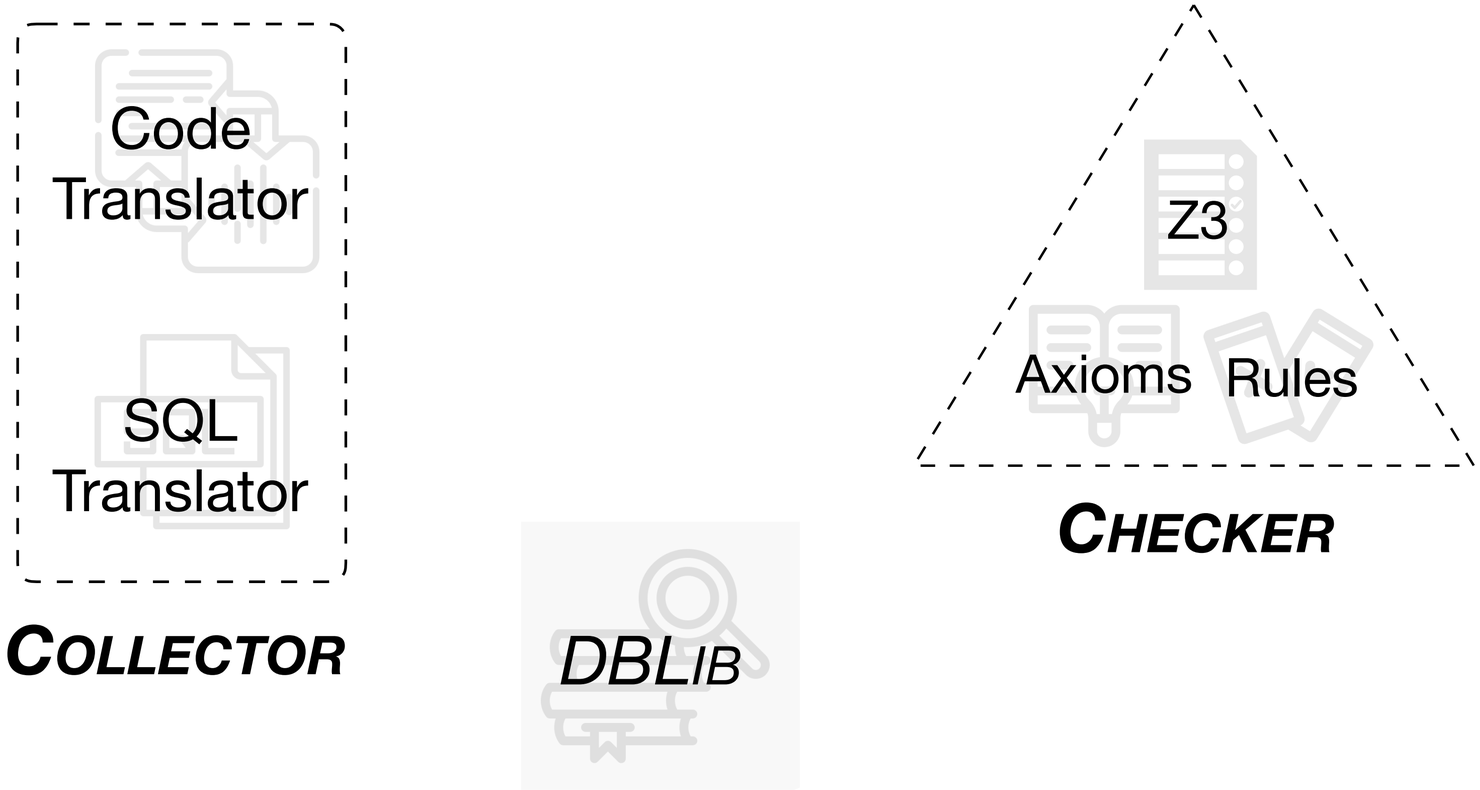
AutoGR



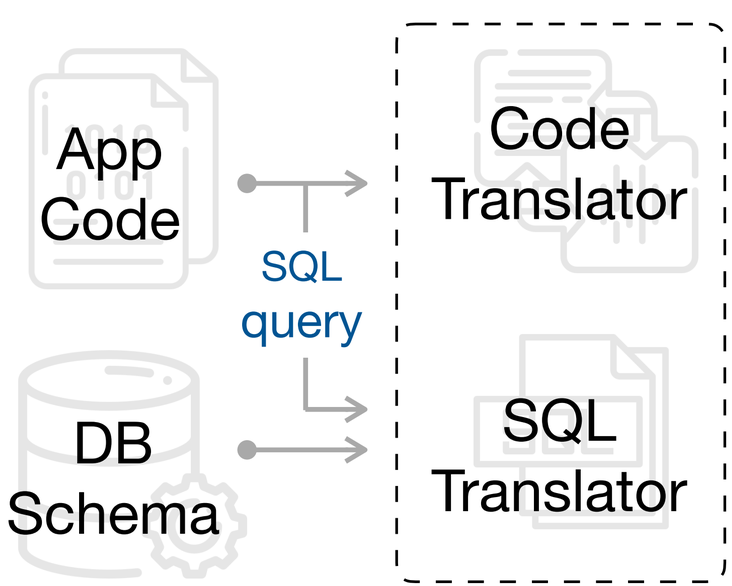
The static analyzer Rigi identifies a minimal set of ordering restrictions that must be ensured so that the intended semantics are not violated.

AutoGR leverages on an existing geo-replication framework Olisipo that enables fine-grained co-ordination over pairs of operations that produce conflicting side effects.

AutoGR —— Rigi

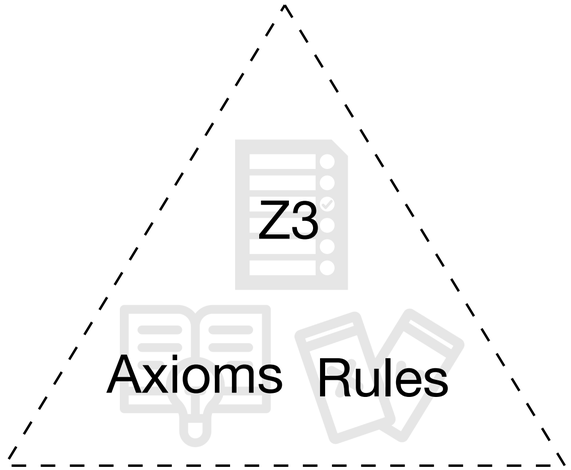

Code
Translator
SQL
Translator
COLLECTOR
DBLIB
Z3
Axioms
Rules
CHECKER

AutoGR —— Rigi



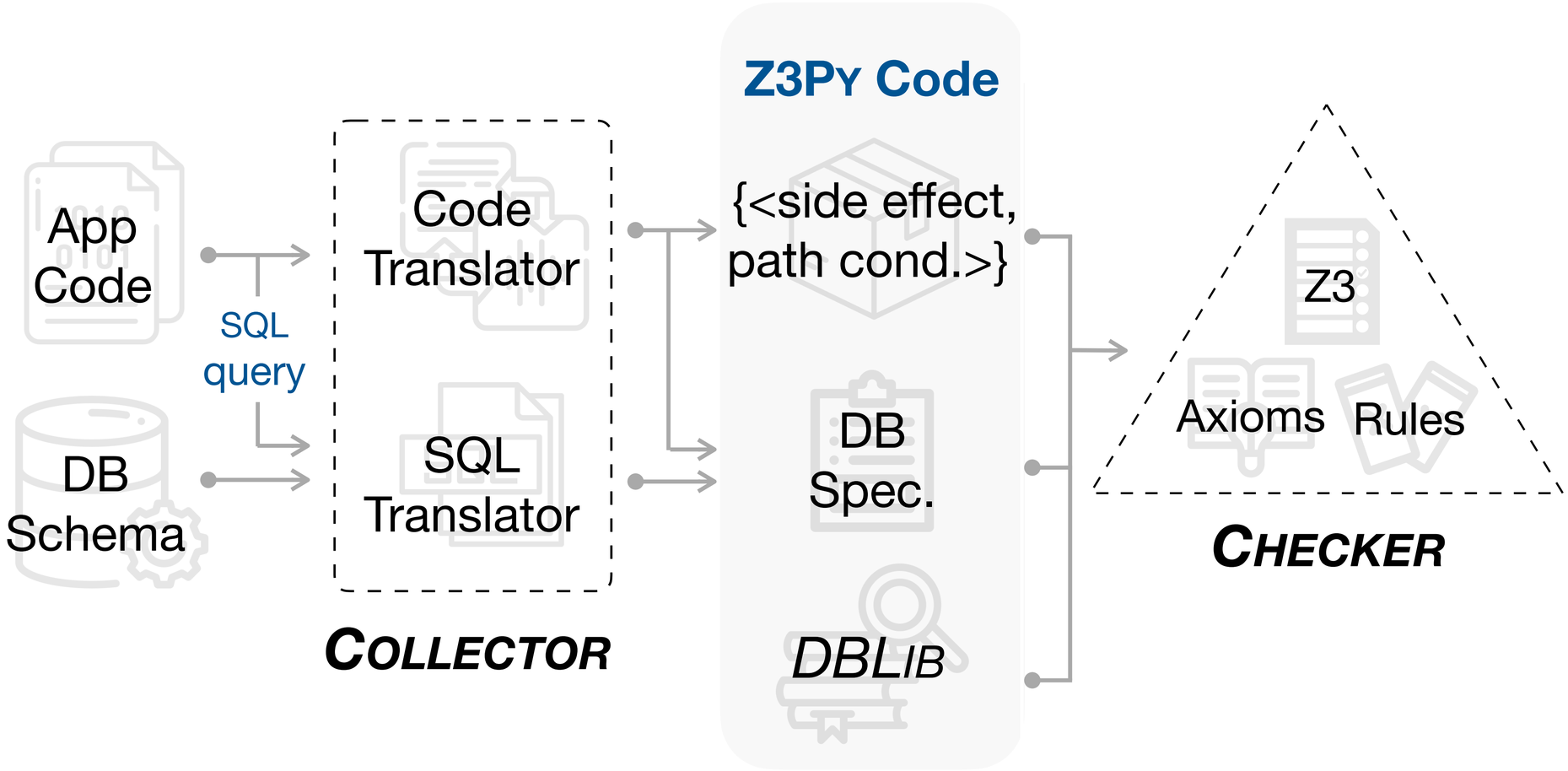
COLLECTOR

DBLIB

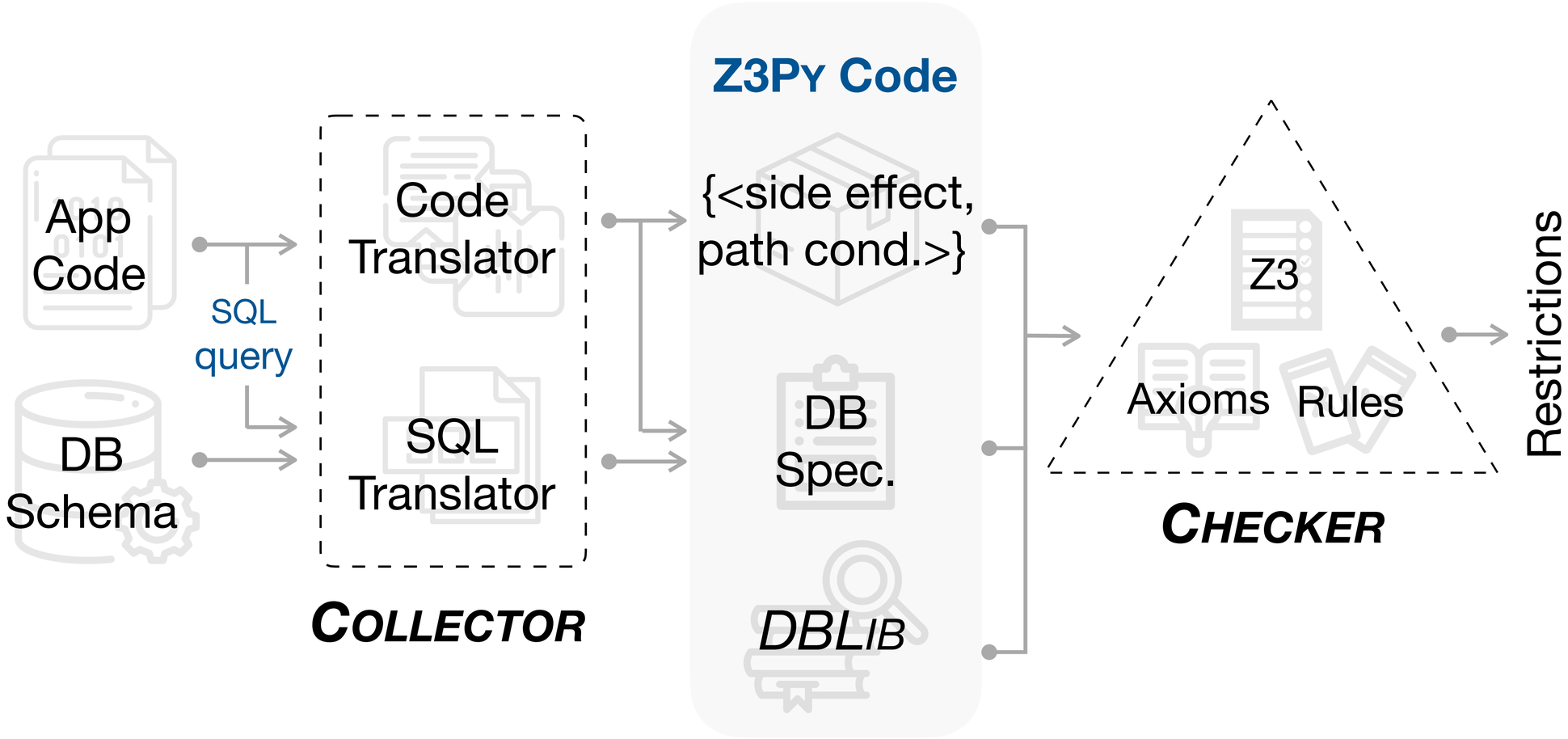


CHECKER

AutoGR —— Rigi


App Code
DB Schema
SQL query
Code Translator
SQL Translator
COLLECTOR
Z3Py Code
{<side effect, path cond.>}
DB Spec.
DBLIB
Z3
Axioms
Rules
CHECKER

AutoGR —— Rigi


App Code
DB Schema
SQL query
Code Translator
SQL Translator
COLLECTOR
Z3Py Code
{<side effect, path cond.>}
DB Spec.
DBLIB
Z3
Axioms
Rules
CHECKER
Restrictions

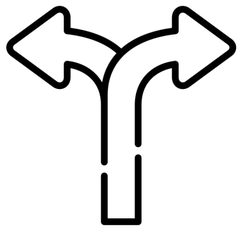
AutoGR –– Rigi –– DBLib

```
CREATE TABLE RSVN (
  R_C_ID  BIGINT NOT NULL,
  R_F_ID  BIGINT NOT NULL,
  R_SEAT  BIGINT NOT NULL, ...
PRIMARY KEY (R_C_ID,R_F_ID))
```

```
UPDATE CUSTOMER
  SET C_BASE_AP_ID = aid,
  LOCATION = loc
  WHERE C_ID = cid
```

Table definition interface

- Primary key, foreign key
- Key with multiple fields



SQL query interface

- Basic: select, update, …
- Advanced: comparison, inner join, aggregation, …

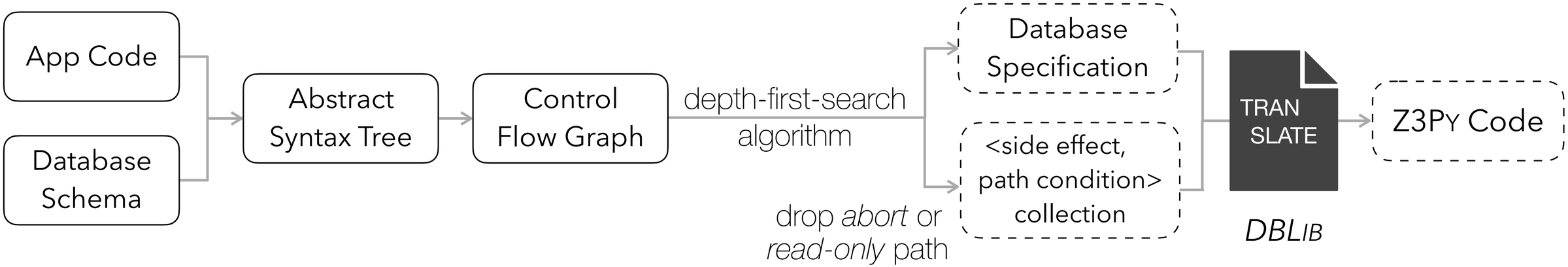
Z3Py

- IntSort, StringSort, BoolSort, RealSort, …
- Extensional array
- ForAll, Implies, <=, ==, …

```
K_RSVN = Datatype(...)
K_RSVN.declare(...,(R_C_ID, R_F_ID))
V_RSVN = Datatype(...)
V_RSVN.declare(...)
TABLE_RSVN = Array(..., K_RSVN, V_RSVN)
```

```
Store(TABLE_CUSTMER, T_CUSTMER.new(cid),
  V_CUSTMER.new(V_CUSTMER.BALANCE(
  Select(TABLE_CUSTMER,T_CUSTMER.new(cid)))
  ,aid,...,loc,...))
```

AutoGR —— Rigi —— Collector



Optimizations:

- CRDTs support

  e.g., “Last-Writer-Win (LWW)” strategy for merging concurrent updates.

- Uniqueness

  Support database’s AUTOINCREMENTAL feature.

AutoGR —— Rigi —— Checker

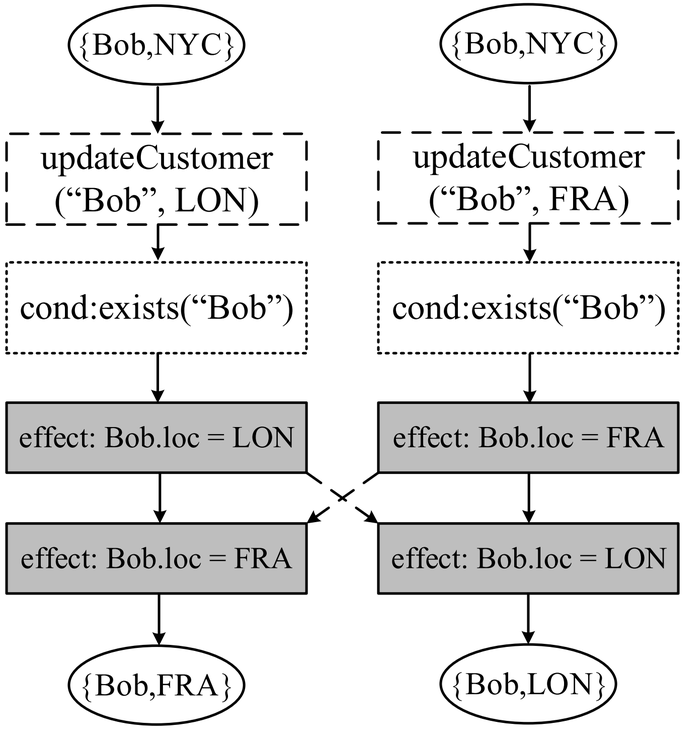
Commutativity check: *Check the commutativity of side-effects to ensure the convergence of the system.*

Semantics check: *If the side-effect of operation A can be generated without seeing the side-effect of operation B, then it must be able to be generated when seeing the side-effect of operation B.*

AutoGR —— Rigi —— Checker

Commutativity check: *Check the commutativity of side-effects to ensure the convergence of the system.*

Semantics check: *If the side-effect of operation A can be generated without seeing the side-effect of operation B, then it must be able to be generated when seeing the side-effect of operation B.*

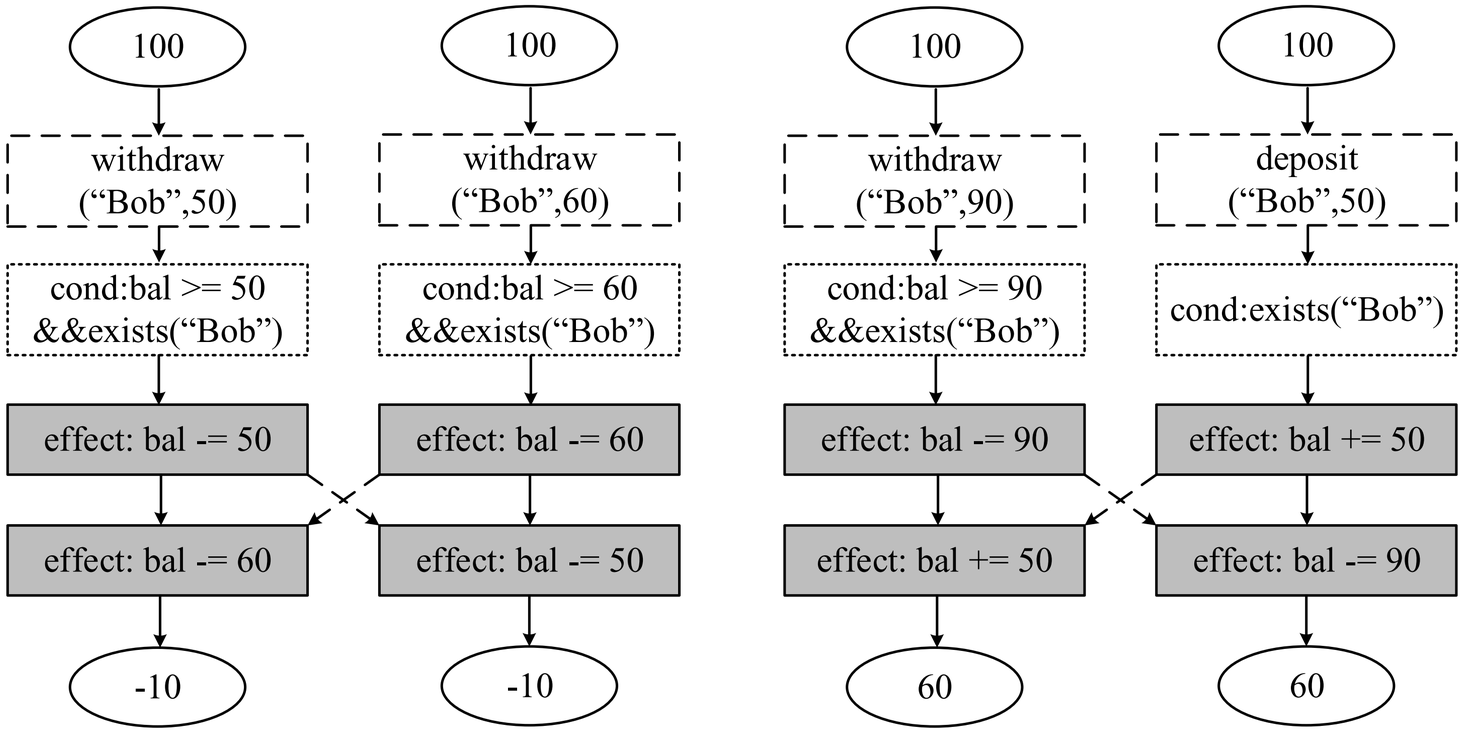


(a) Violating execution 1

AutoGR —— Rigi —— Checker

Commutativity check: *Check the commutativity of side-effects to ensure the convergence of the system.*

Semantics check: *If the side-effect of operation A can be generated without seeing the side-effect of operation B, then it must be able to be generated when seeing the side-effect of operation B.*



(b) Violating execution 2 **(c) Correct execution.**

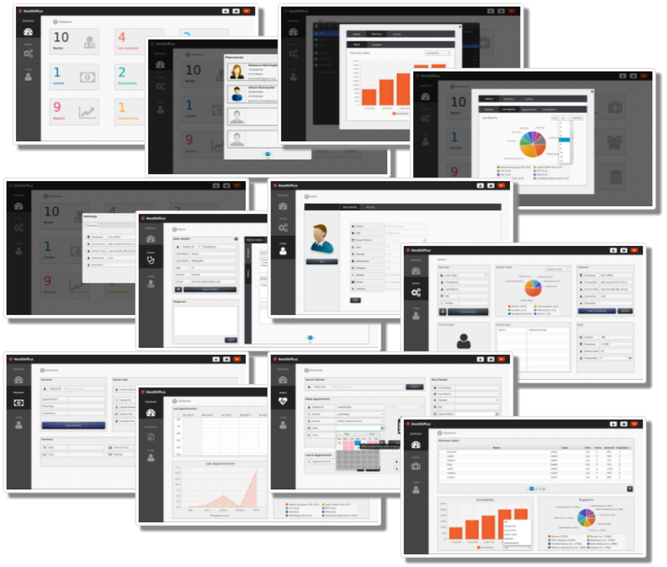
Case Study

- *SmallBank (codebase 2.5k, 5 transactions):*
  Simulating an online banking system.

- *RUBiS (codebase 9.8k, 16 transactions):*
  An eBay-like online auction website.

- *SeatsReservation (codebase 5.0k, 6 transactions):*
  An electronic airline ticketing service.

- *HealthPlus (codebase 15.7k, 157 transactions):*
  A real-world deployable management system for health care facility.



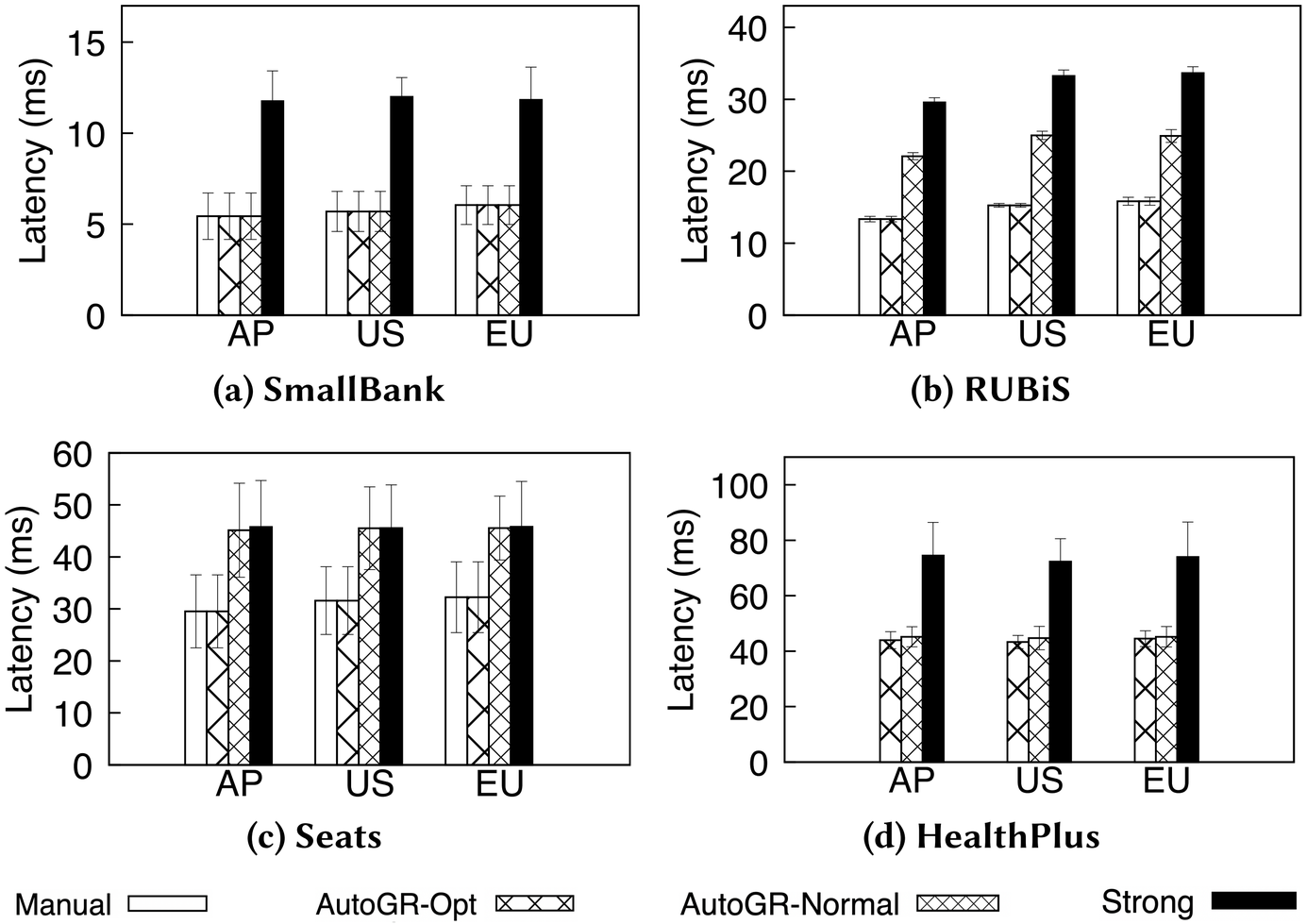
Case Study —— Static analysis

	Lines of Z3Py Code Generated by Rigi			Analysis Cost	Restriction Rate (Normal / Opt)
	Database	Path condition	Side effect		
SmallBank	29	38	119	~ 24s	20% / 20%
RUBiS	113	62	191	~ 3.4min	23% / 9%
Seats	267	65	207	~ 6.7min	39% / 31%
HealthPlus	524	1113	1387	~ 1.5h (~ 7.7min for 16 threads)	2.9% / 1.4%

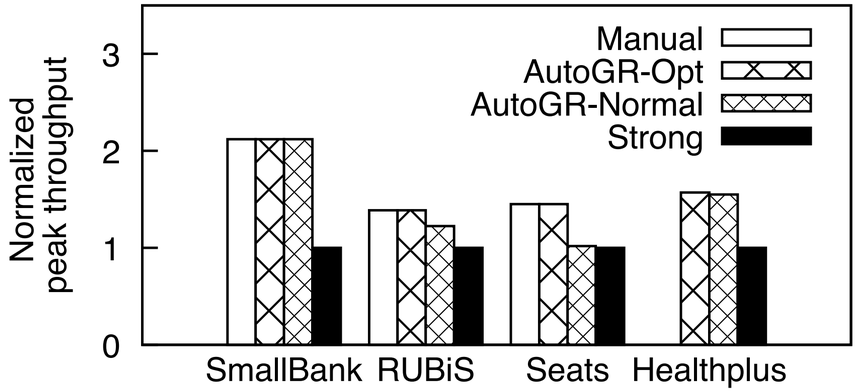
Considering that the analysis is a one-time and offline job, the cost is moderate.

Case Study —– Geo-replication

Average latency perceived by users



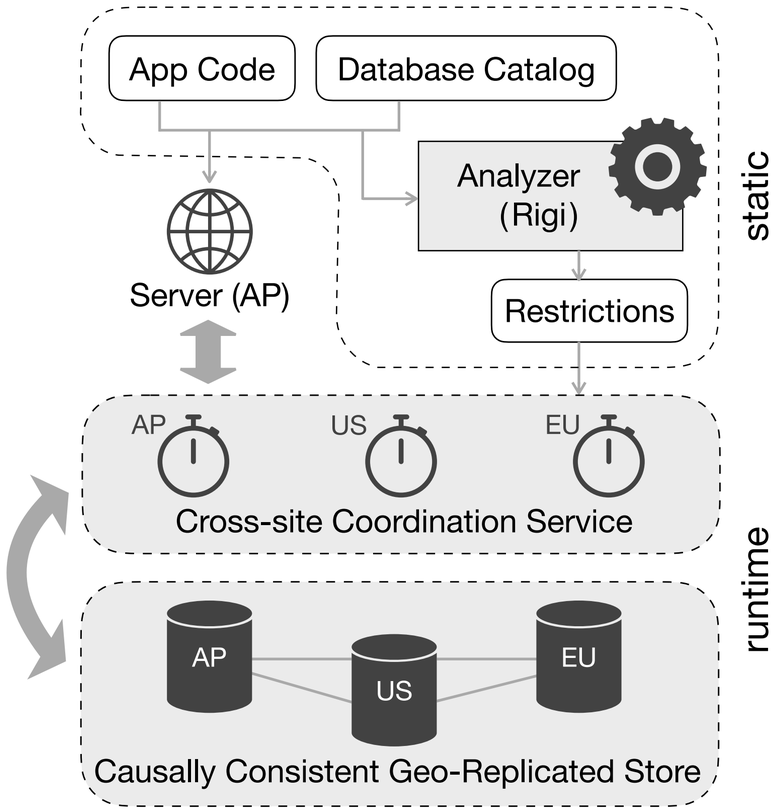
Normalized peak throughput numbers

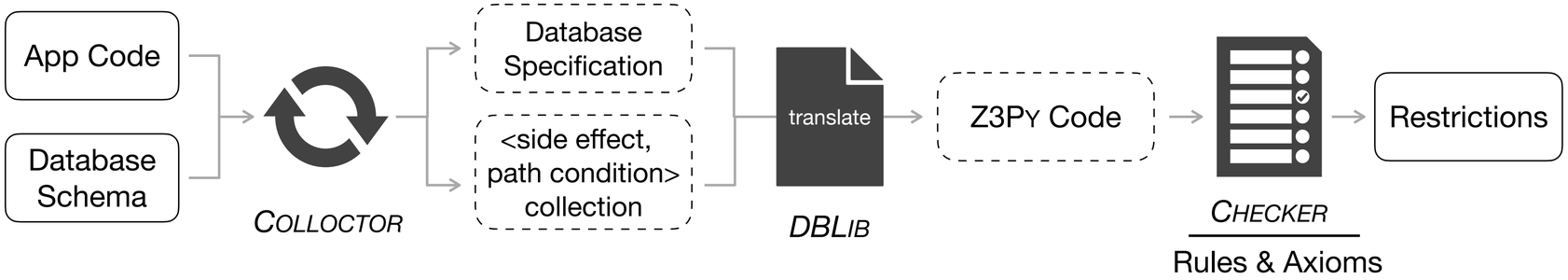


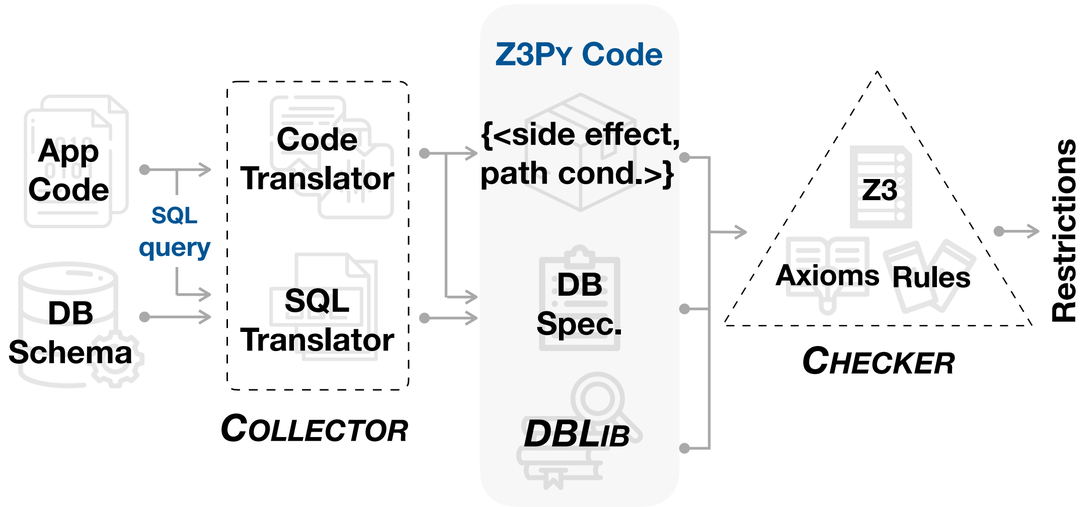
- *Compare with human-intervention-free automated approaches: reduces up to 61.8% latency and achieves up to 2.12× higher peak throughput.*

- *Compare with manual analysis approaches: quickly enable the geo-replication feature with zero human intervention while offering similarly low latency and high throughput.*

Thank You !


App Code
Database Catalog
Analyzer
(Rigi)
Server (AP)
Restrictions
static
AP
US
EU
Cross-site Coordination Service
runtime
AP
US
EU
Causally Consistent Geo-Replicated Store

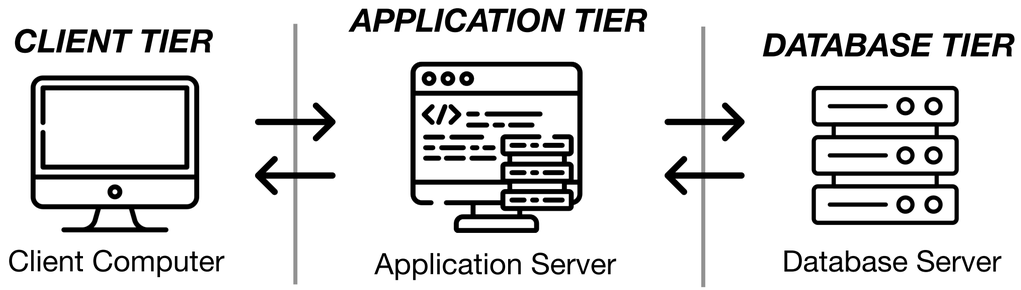

App Code
Database Schema
COLLOCTOR
Database Specification
<side effect, path condition> collection
translate
DBLIB
Z3PY Code
CHECKER
Rules & Axioms
Restrictions

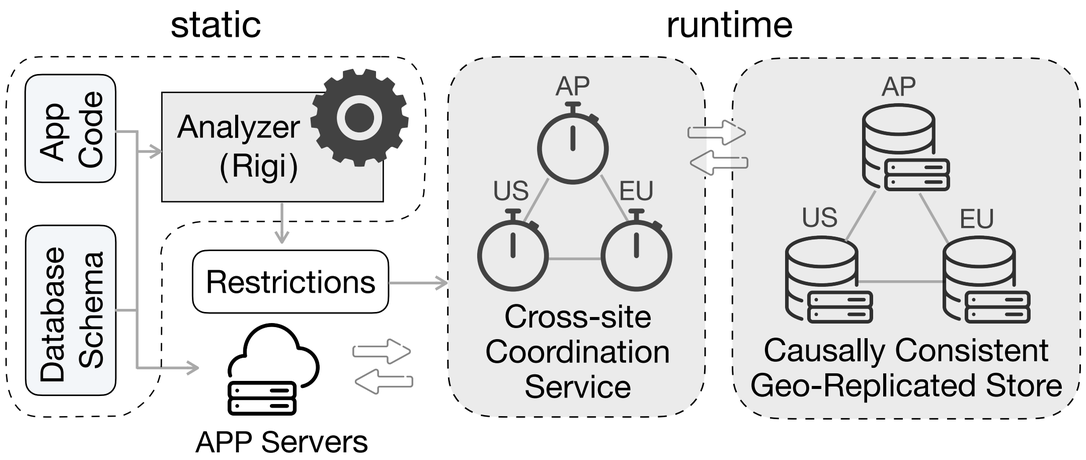

Z3Py Code
App Code
DB Schema
SQL query
Code Translator
SQL Translator
COLLECTOR
{<side effect, path cond.>}
DB Spec.
DBLIB
Z3
Axioms
Rules
CHECKER
Restrictions

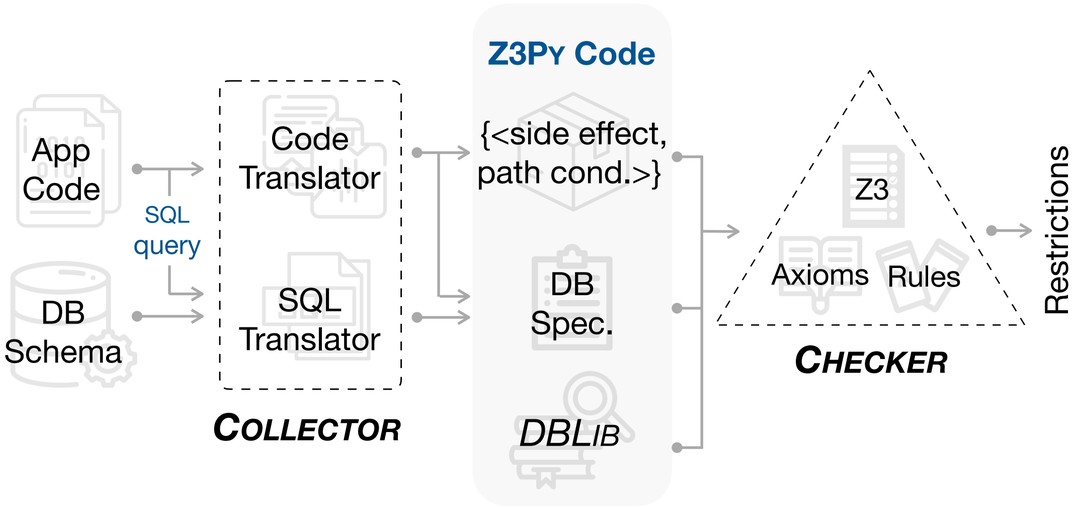

Manual
AutoGR-Opt
AutoGR-Normal
Strong


Single-Site-Strong
3-Site-AutoGR-Optimal
3-Site-AutoGR-Normal
3-Site-Strong

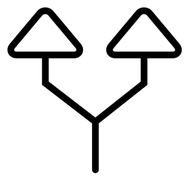

Single-Site-Strong
3-Site-PoR
3-Site-AutoGR-Optimal
3-Site-AutoGR-Normal
3-Site-Strong

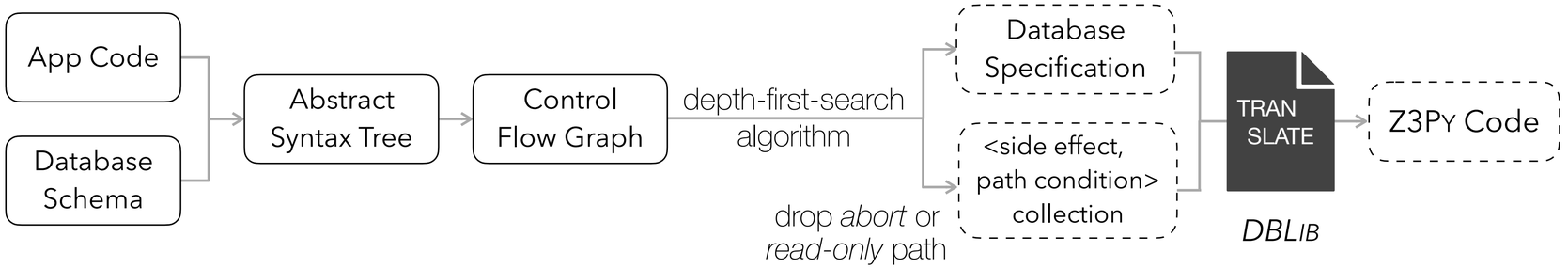

CLIENT TIER
APPLICATION TIER
DATABASE TIER
Client Computer
Application Server
Database Server


static
runtime
App Code
Database Schema
Analyzer (Rigi)
Restrictions
APP Servers
AP
US
EU
Cross-site Coordination Service
AP
US
EU
Causally Consistent Geo-Replicated Store


App Code
DB Schema
SQL query
Code Translator
SQL Translator
COLLECTOR
Z3Py Code
{<side effect, path cond.>}
DB Spec.
DBLIB
Z3
Axioms
Rules
CHECKER
Restrictions

$$
\begin{array}{lcll}
\text{(Operation)} & u & \in & \text{SideEffect} \times \text{PathCondition} \\
\text{(SideEffect)} & o & \in & \text{State} \to \text{State} \\
\text{(PathCondition)} & pcond & \in & (\text{State} \times \text{Argument}) \to \text{Bool} \\
\text{(Argument)} & arg & := & \ldots \\
\text{(State)} & s & := & \ldots
\end{array}
$$




App Code
Database Schema
Abstract Syntax Tree
Control Flow Graph
depth-first-search algorithm
drop *abort* or *read-only* path
Database Specification
<side effect, path condition> collection
TRAN SLATE
DBLIB
Z3PY Code