Automating the Choice of Consistency Levels in Replicated Systems

Cheng Li[†], João Leitão[§], Allen Clement[†]
Nuno Preguiça[§], Rodrigo Rodrigues[§], Viktor Vafeiadis[†]

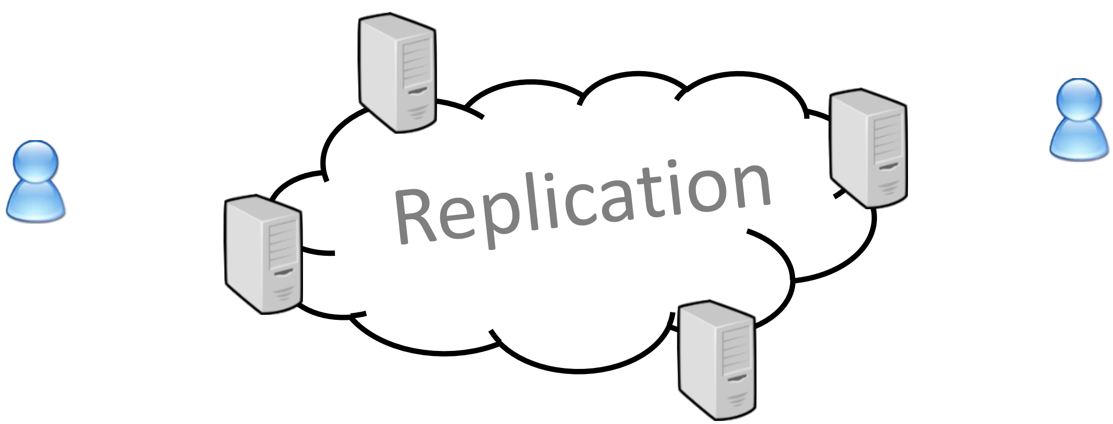
Max Planck Institute for Software Systems (MPI-SWS)[†],
NOVA-LINCS / Universidade Nova de Lisboa[§]

Max Planck Institute for Software Systems

NOVALINCS

RedBlue Consistency [OSDI'12]

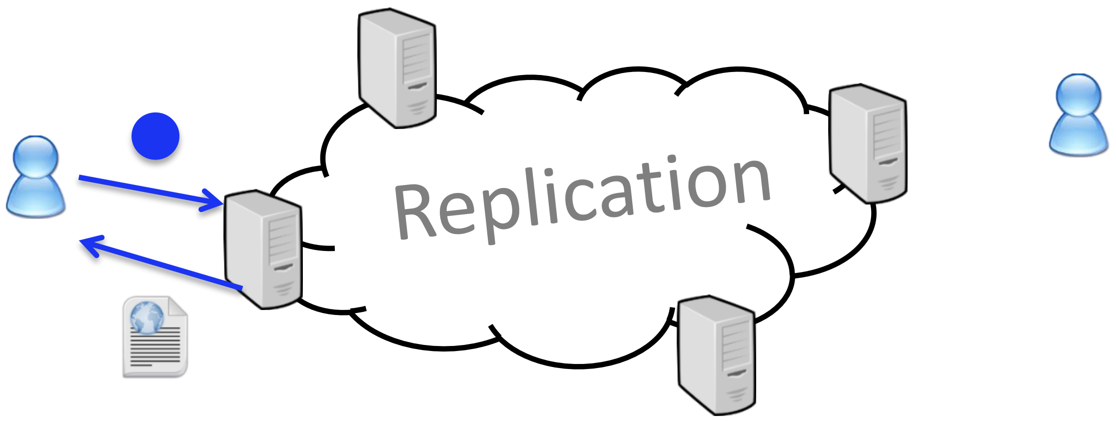
Builds replicated systems that are fast



RedBlue Consistency [OSDI'12]

Builds replicated systems that are fast

Blue ops: local, fast, weakly consistent



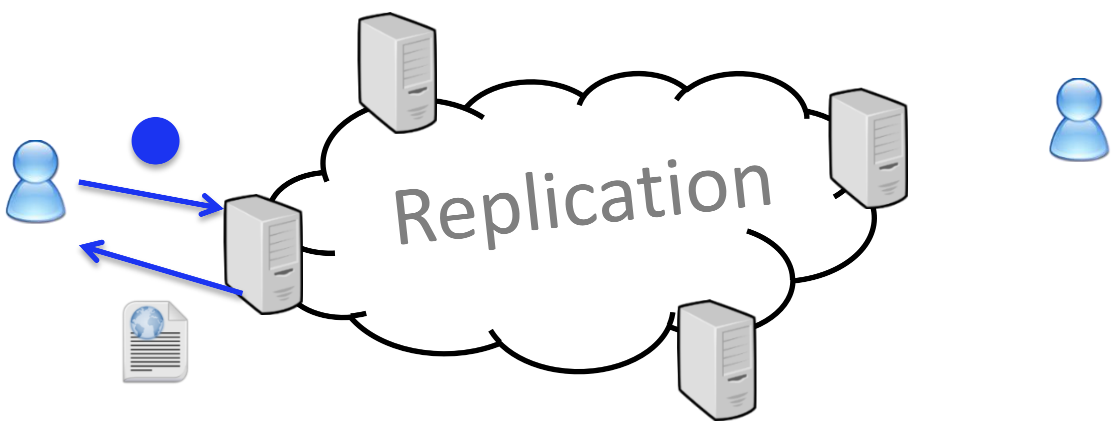
RedBlue Consistency [OSDI'12]

Builds replicated systems that are fast and correct

State convergence

Invariant preservation

Blue ops: local, fast, weakly consistent



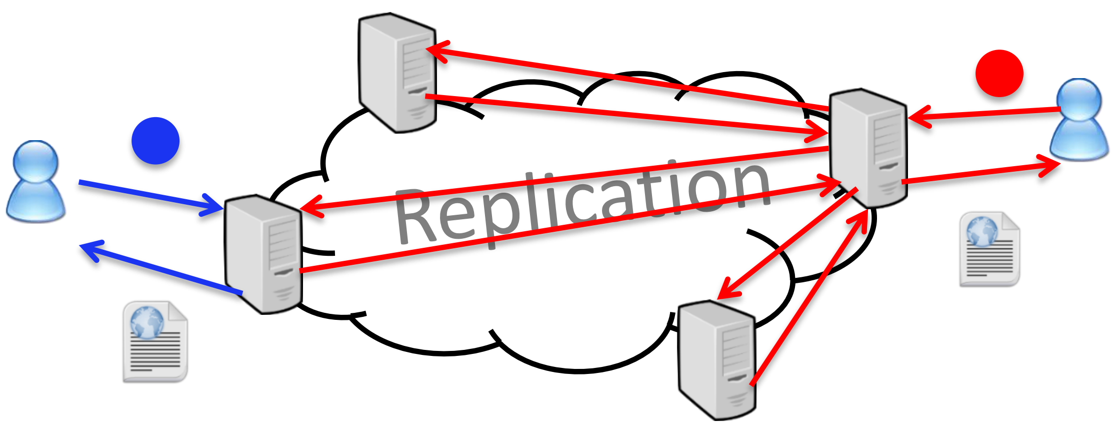
RedBlue Consistency [OSDI'12]

Builds replicated systems that are fast and correct

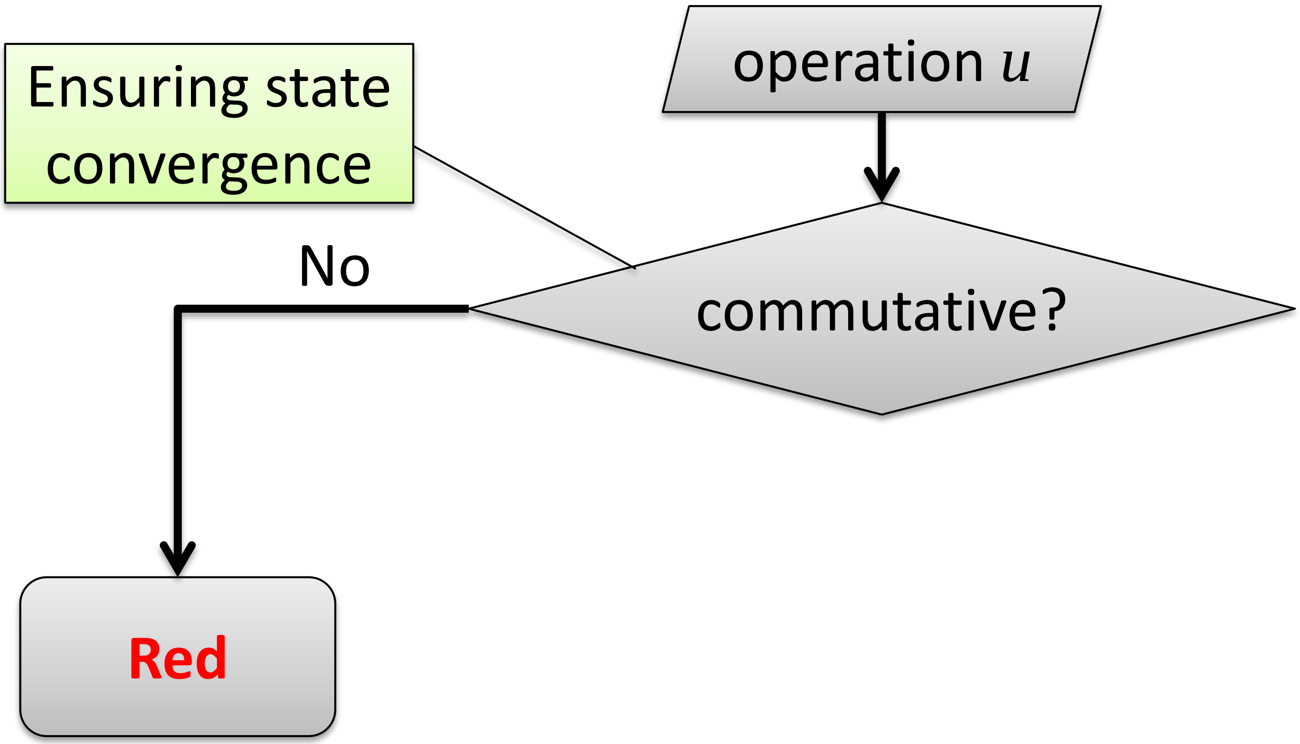
- State convergence
- Invariant preservation

Blue ops: local, fast, weakly consistent

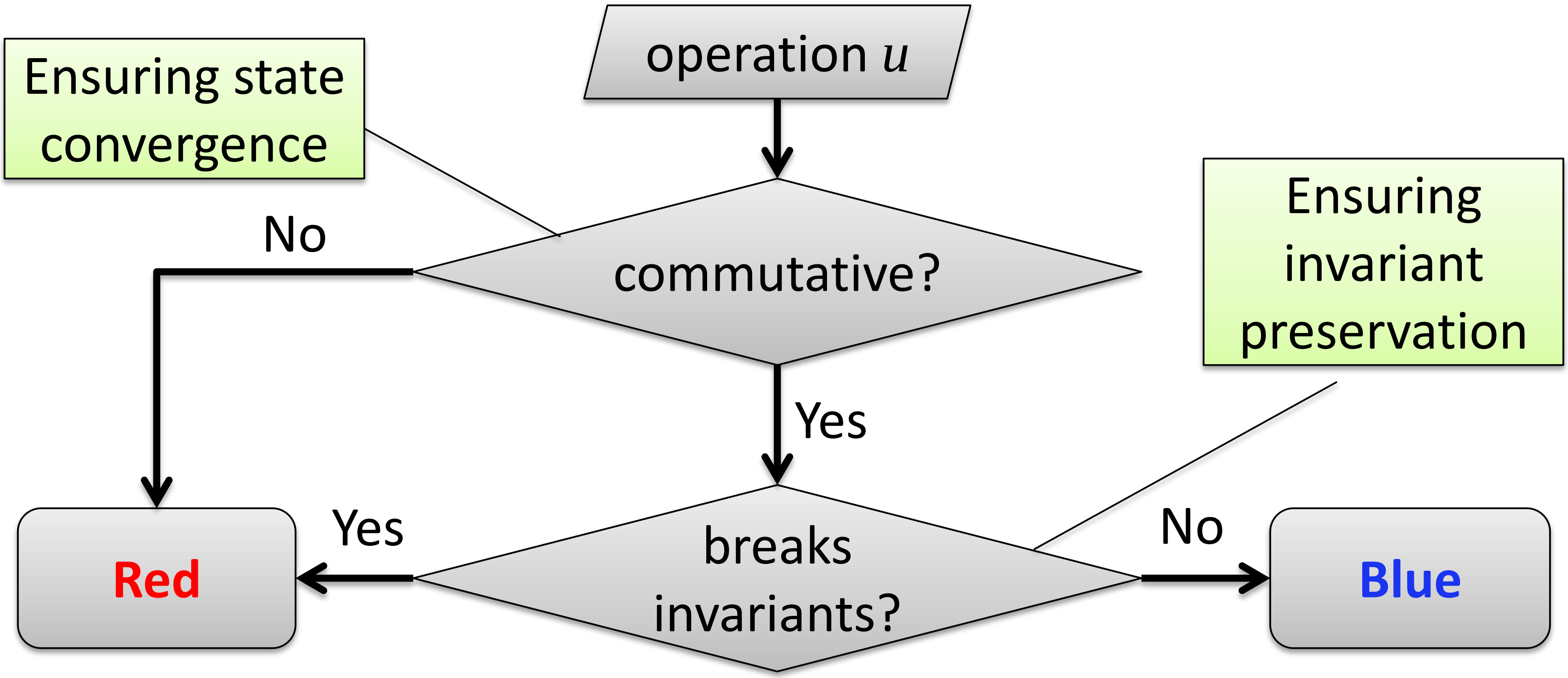
Red ops: global, slow, strongly consistent



Choosing between Blue or Red

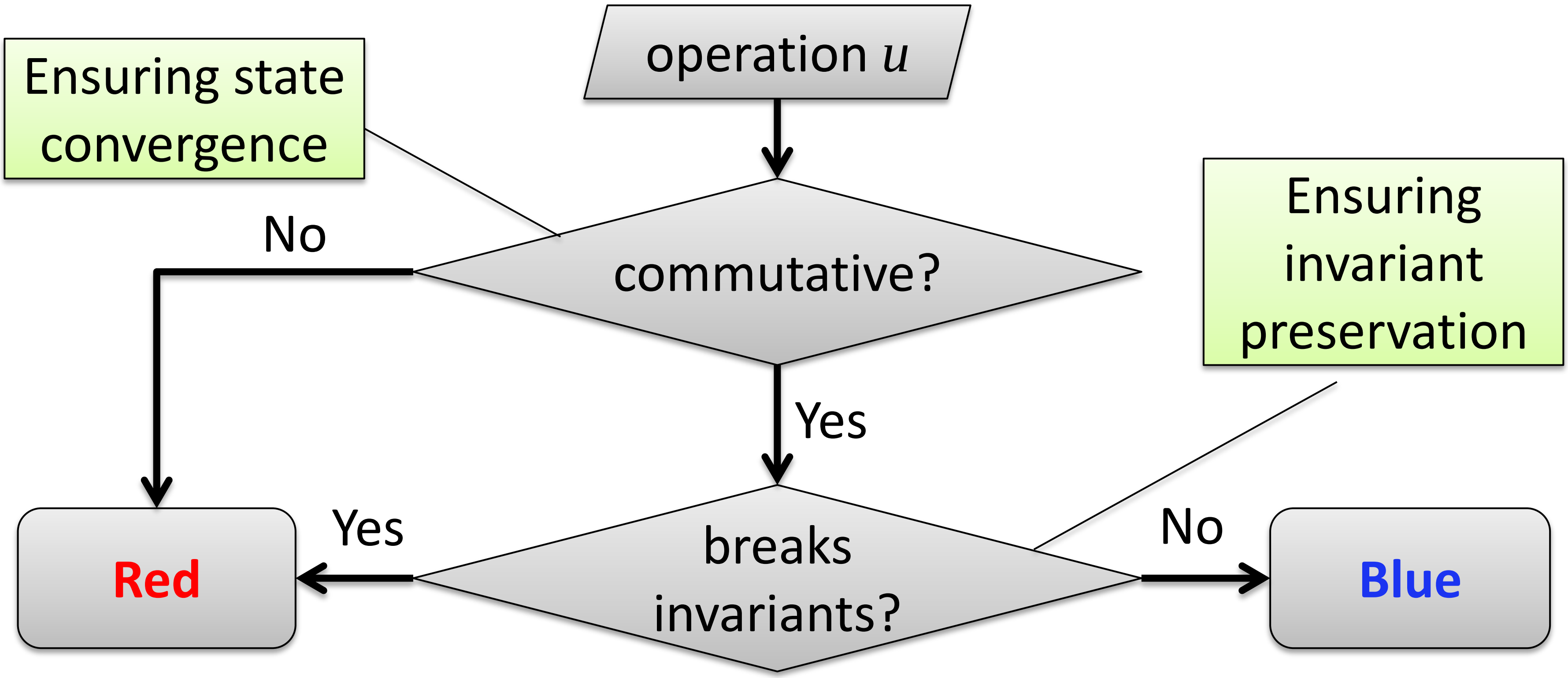


Choosing between Blue or Red

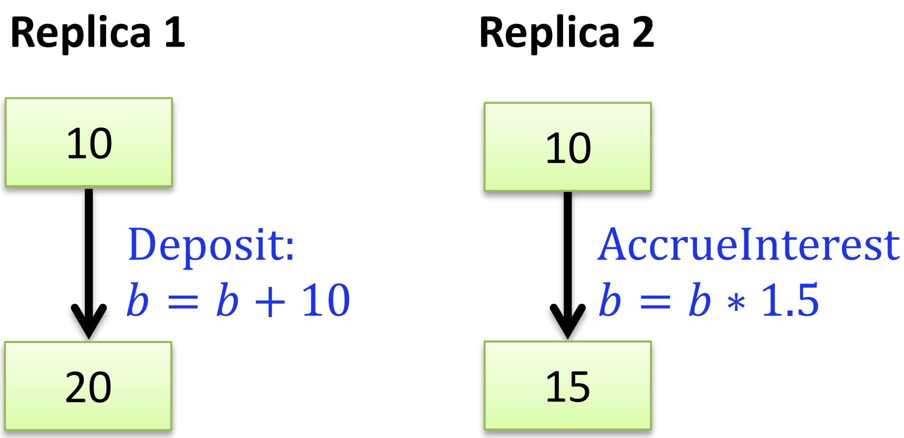


Choosing between Blue or Red

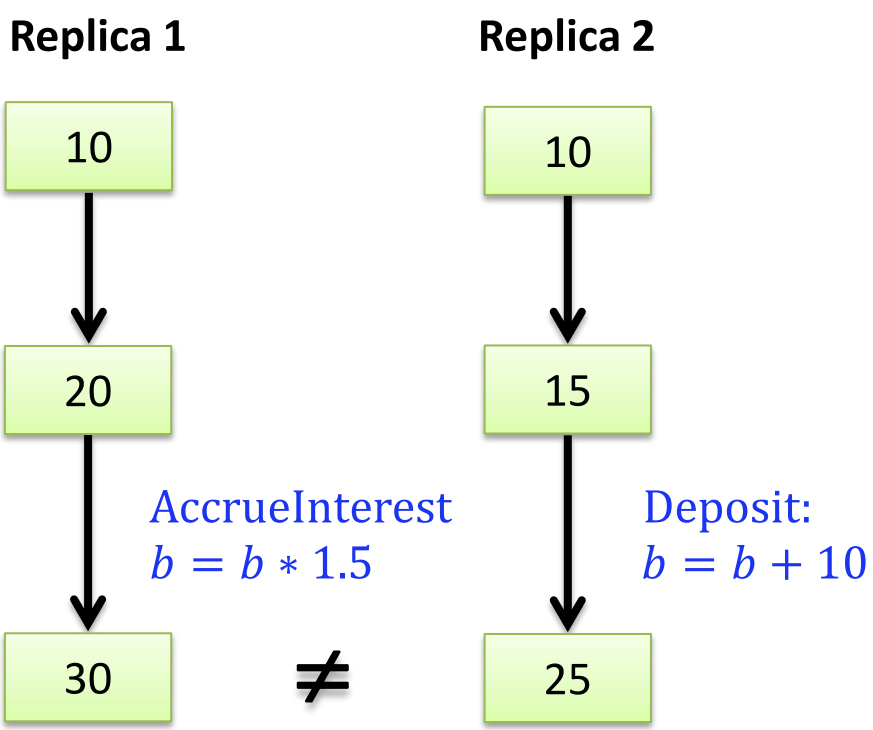
Good performance obtained if blue ops dominate op space



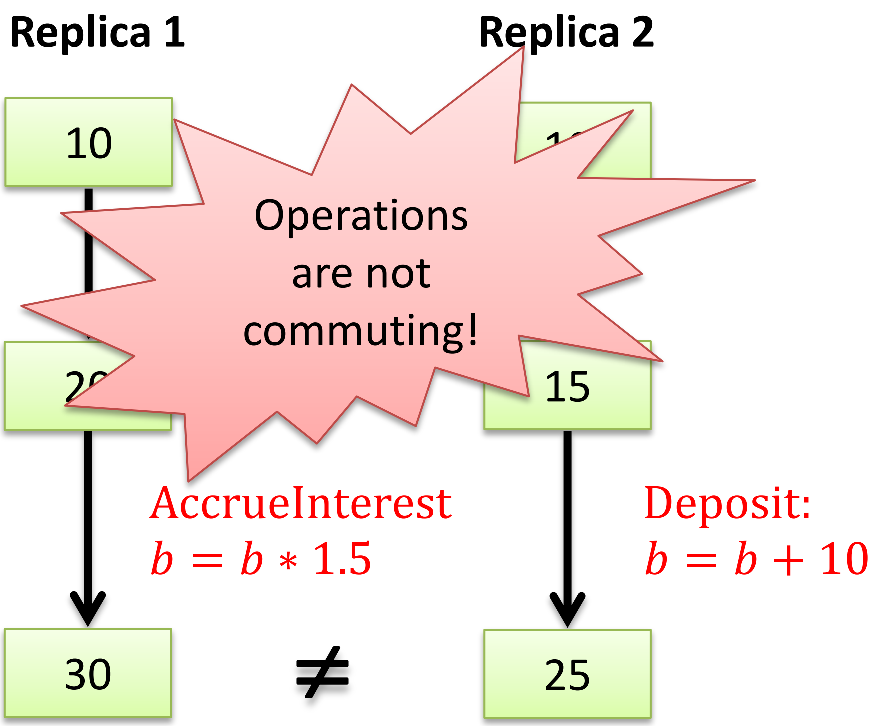
RedBlue Consistent Bank



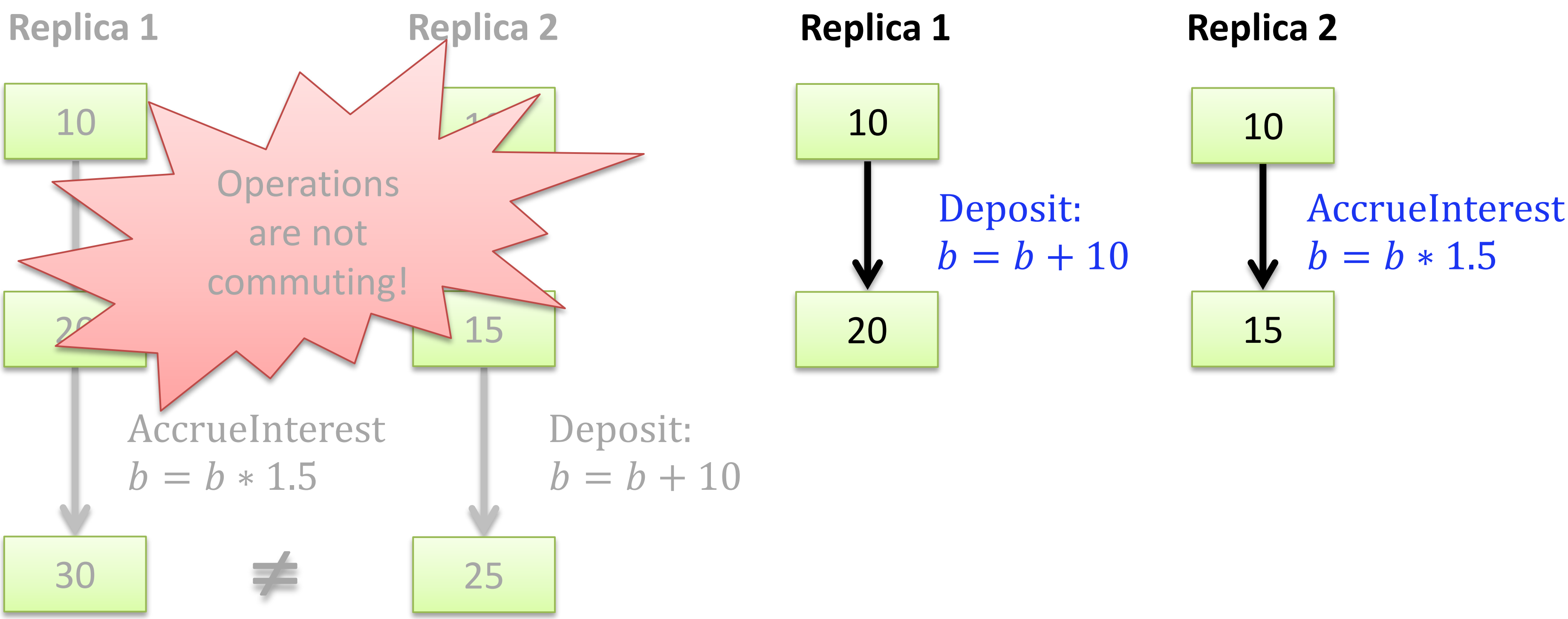
RedBlue Consistent Bank



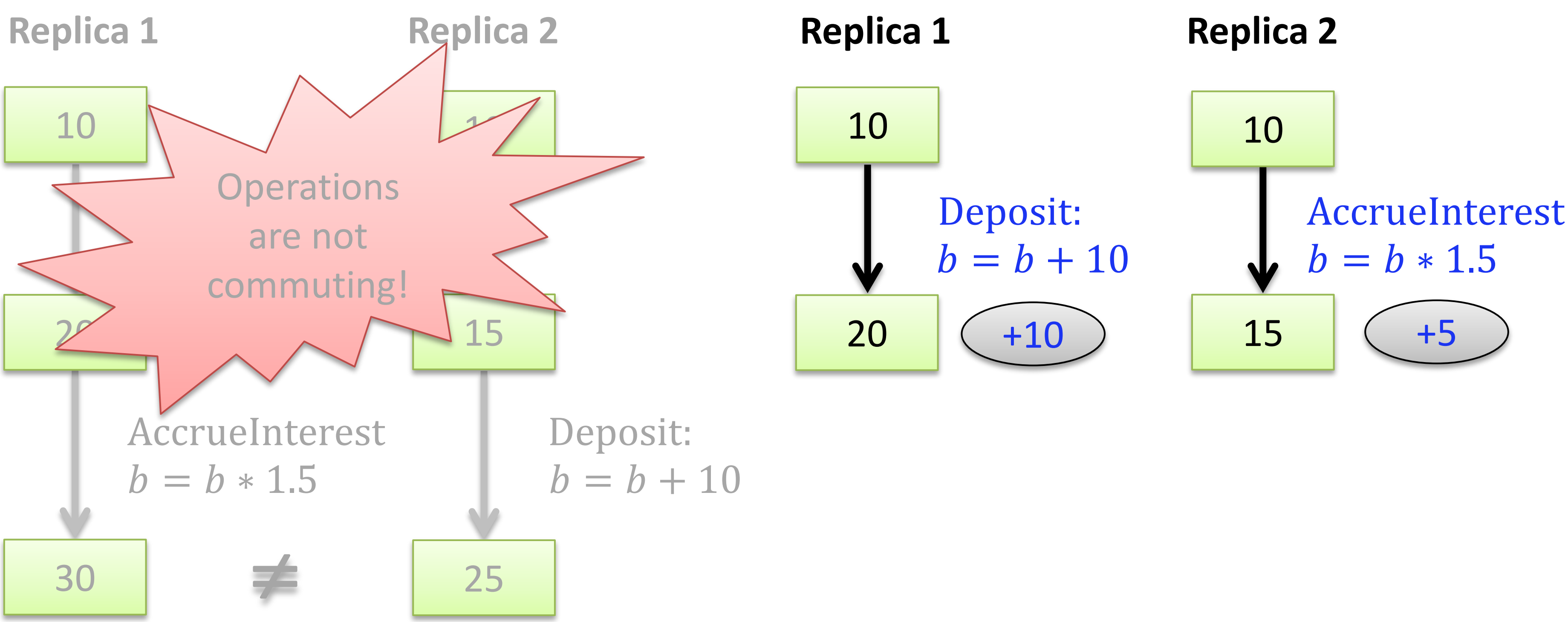
RedBlue Consistent Bank



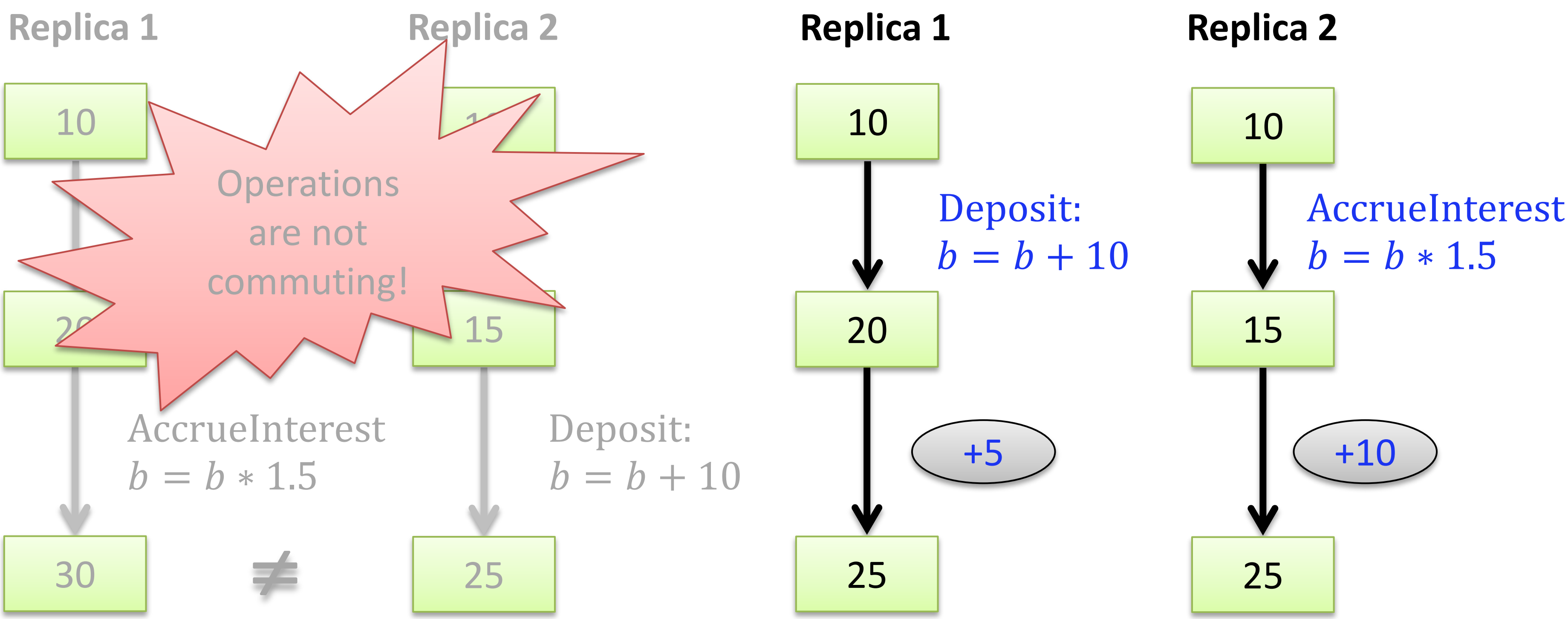
RedBlue Consistent Bank



RedBlue Consistent Bank

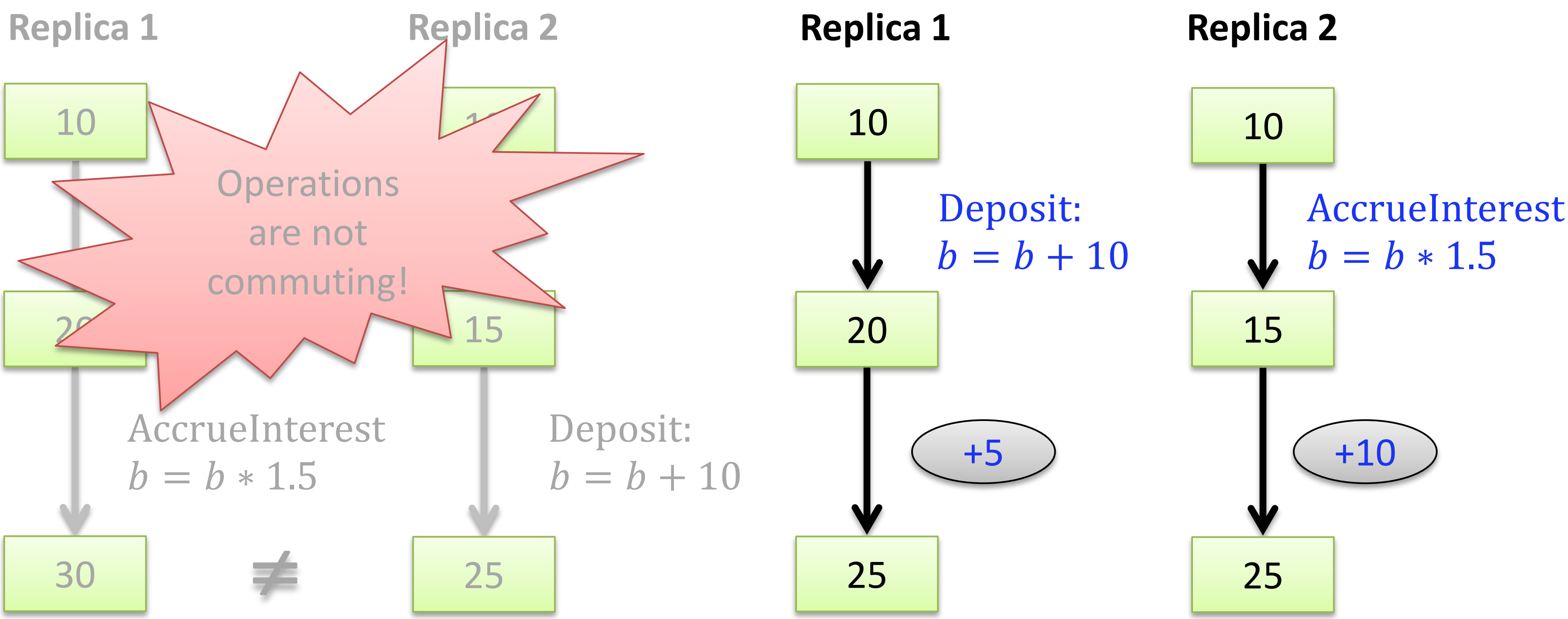


RedBlue Consistent Bank

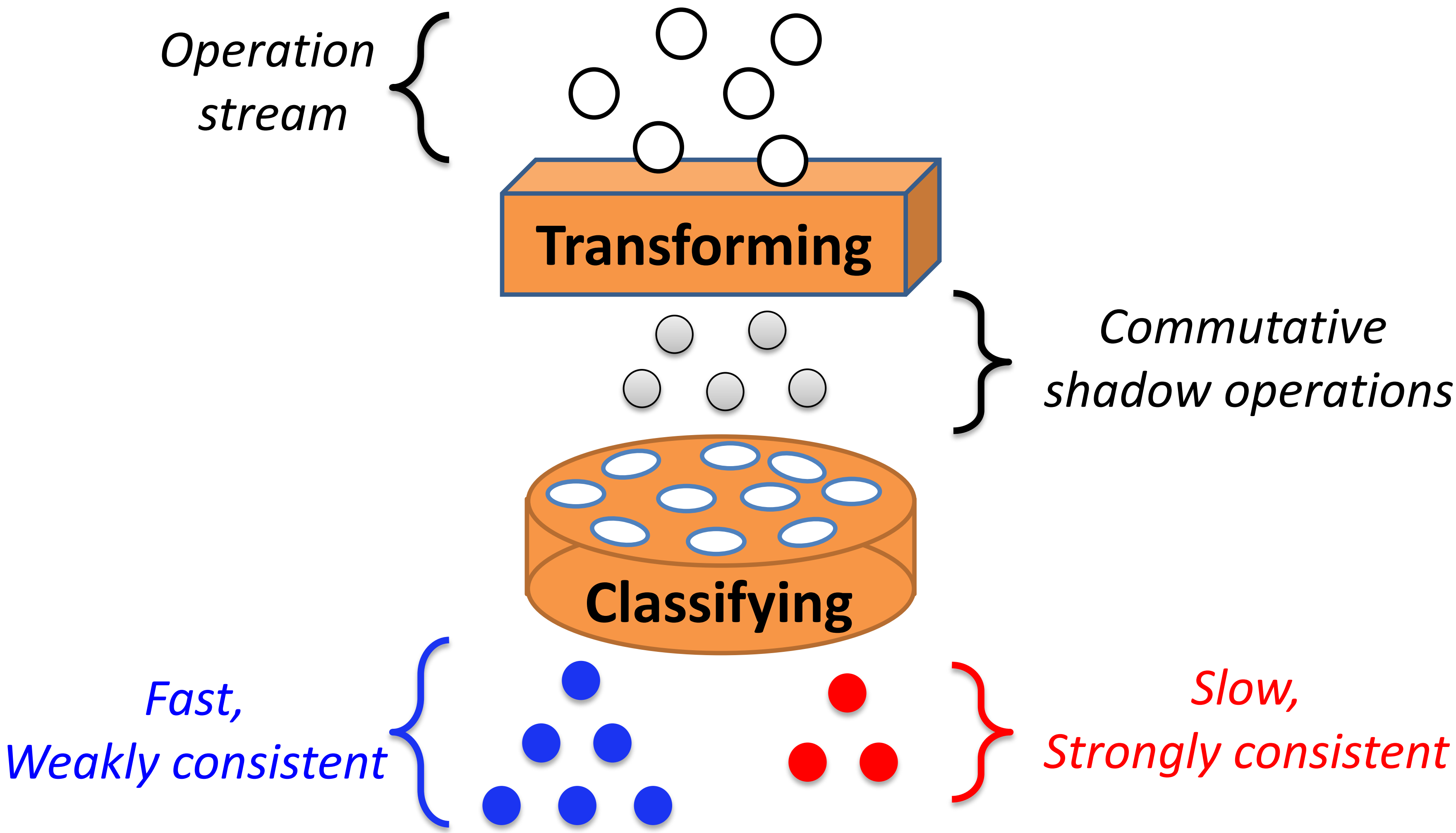


RedBlue Consistent Bank

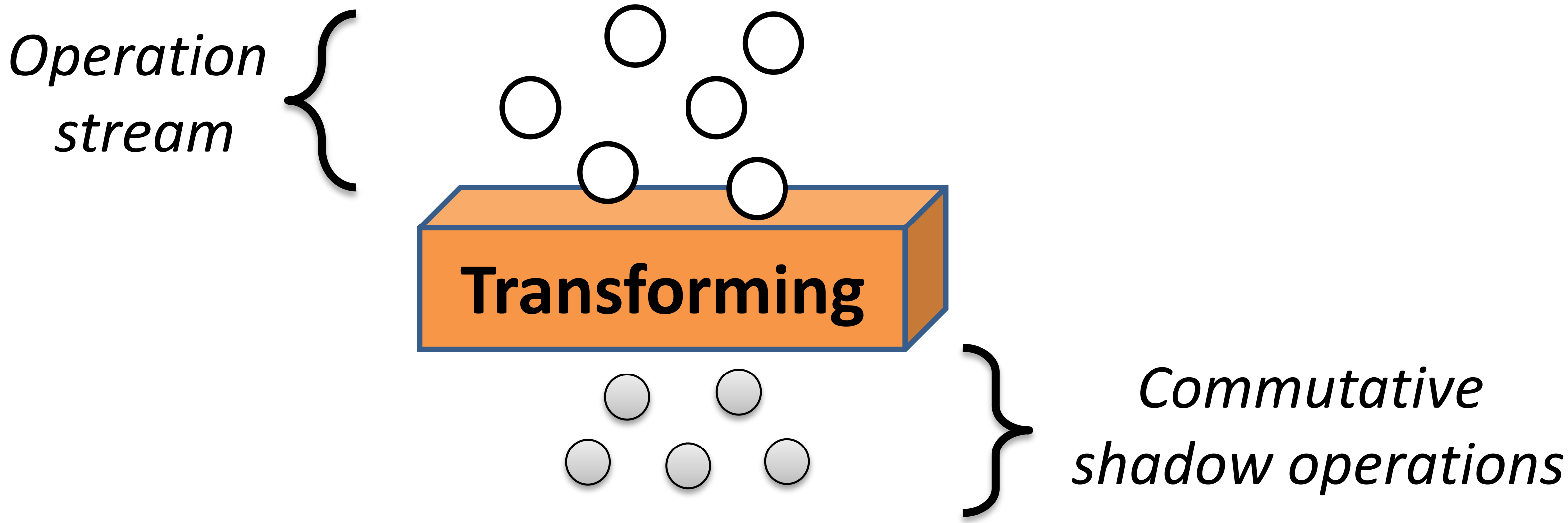
Maximize the blue op space by encoding side effects into commutative **shadow operations**



SIEVE

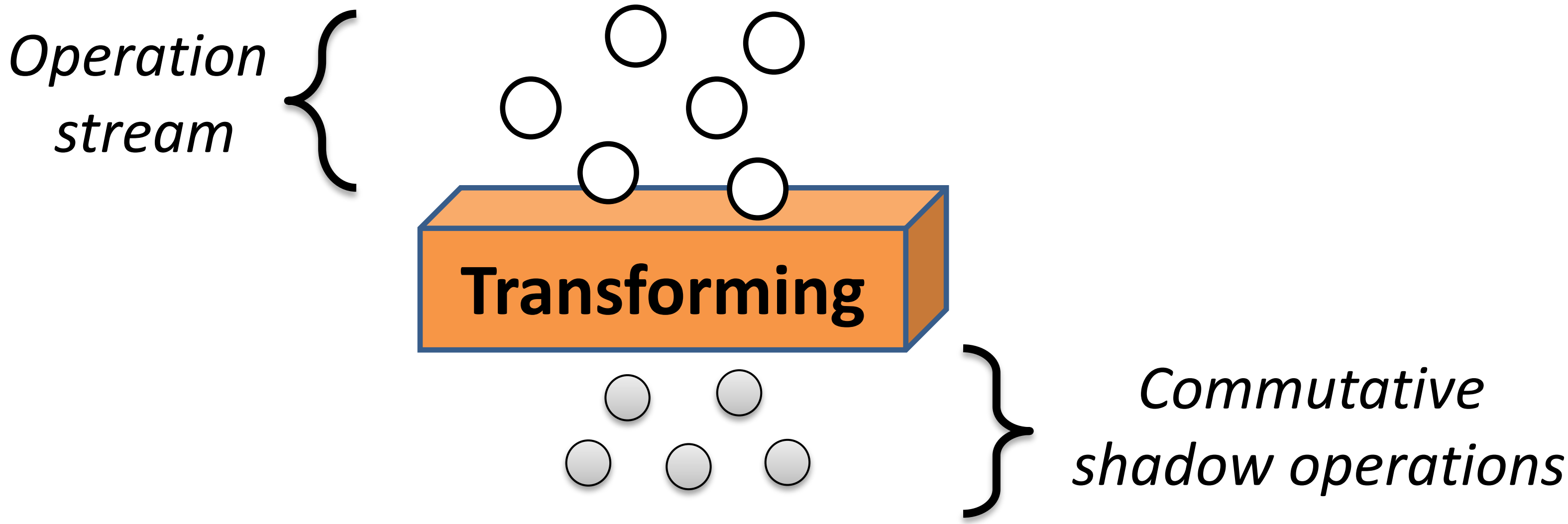

Operation stream
Transforming
Commutative shadow operations
Classifying
Fast, Weakly consistent
Slow, Strongly consistent

SIEVE



Operation stream
Transforming
Commutative shadow operations

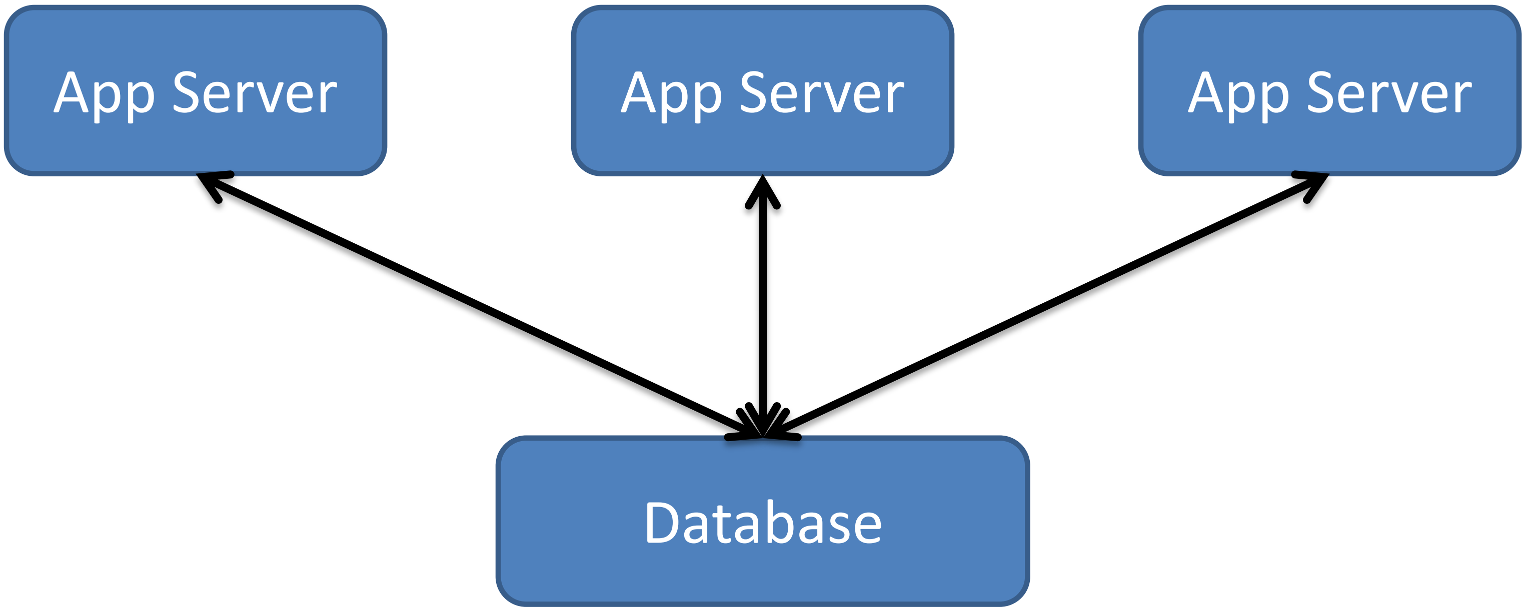
SIEVE



Challenges:

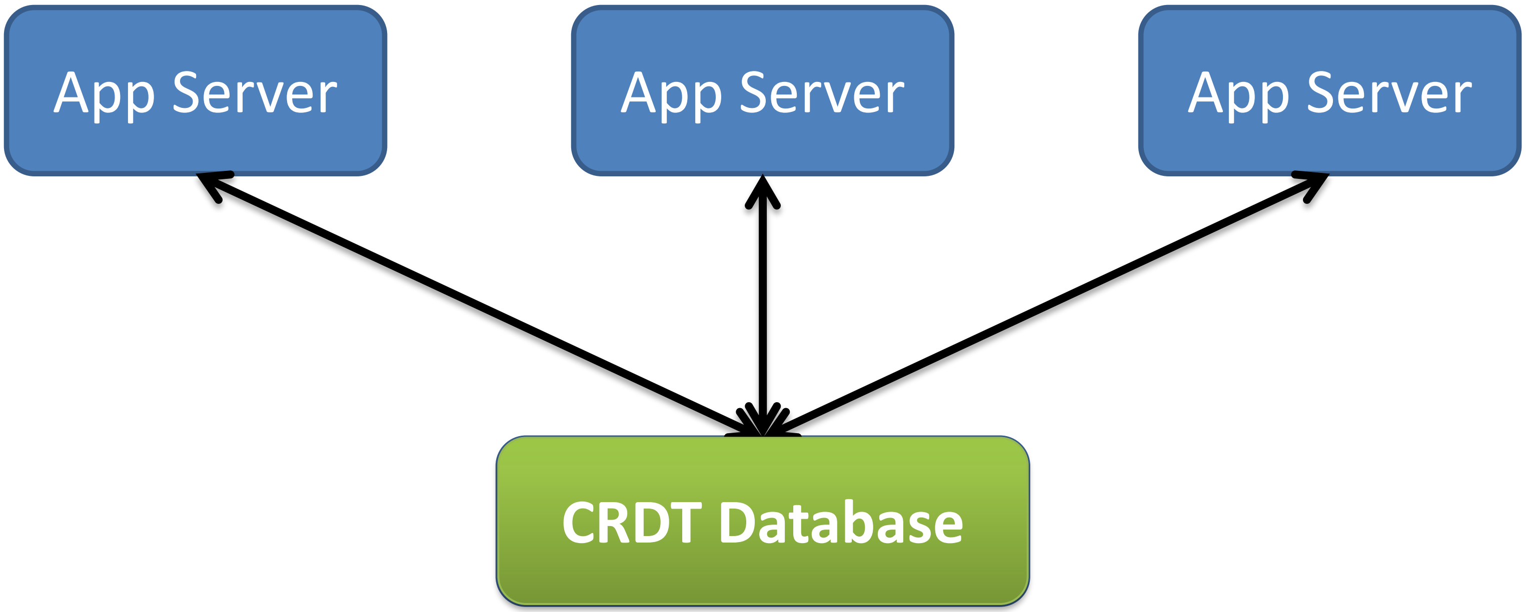
- Making arbitrary side effects commute
- Minimizing human intervention

Two-tier Application Model



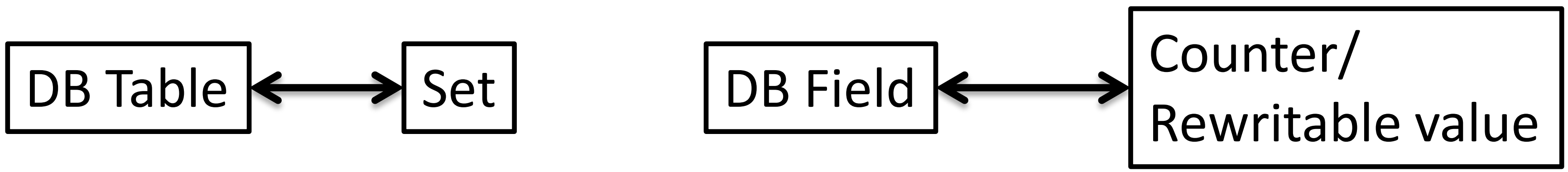
- Observation: Side effects are encapsulated into a sequence of DB statements

Two-tier Application Model

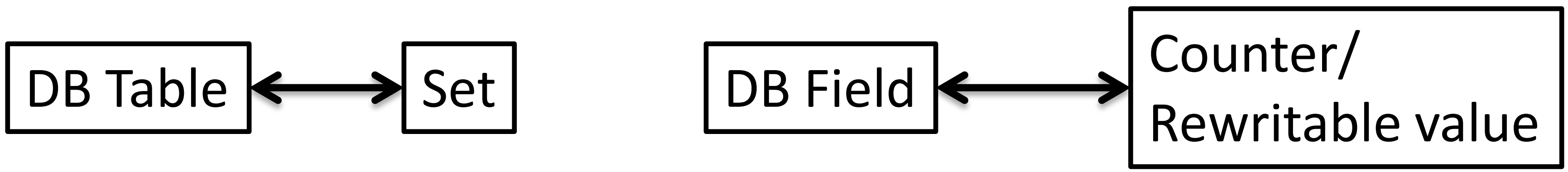


- Observation: Side effects are encapsulated into a sequence of DB statements
- Insight: We can model the database using commutative replicated data types (CRDTs)

Leveraging CRDTs



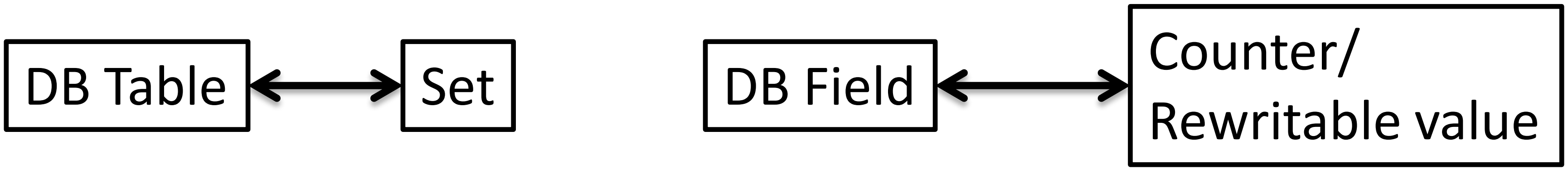
Leveraging CRDTs



- Transform each DB statement into one or more CRDT operations

Shadow operation: [CRDT_OP1; CRDT_OP2; CRDT_OP3;…]

Leveraging CRDTs



- Transform each DB statement into one or more CRDT operations

Shadow operation: [CRDT_OP1; CRDT_OP2; CRDT_OP3;…]

- Programmers ***<u>only</u>*** annotate schema with CRDTs:

@[*CRDTName*][*TableName*|*DataFieldName*]

CRDT Annotation Example

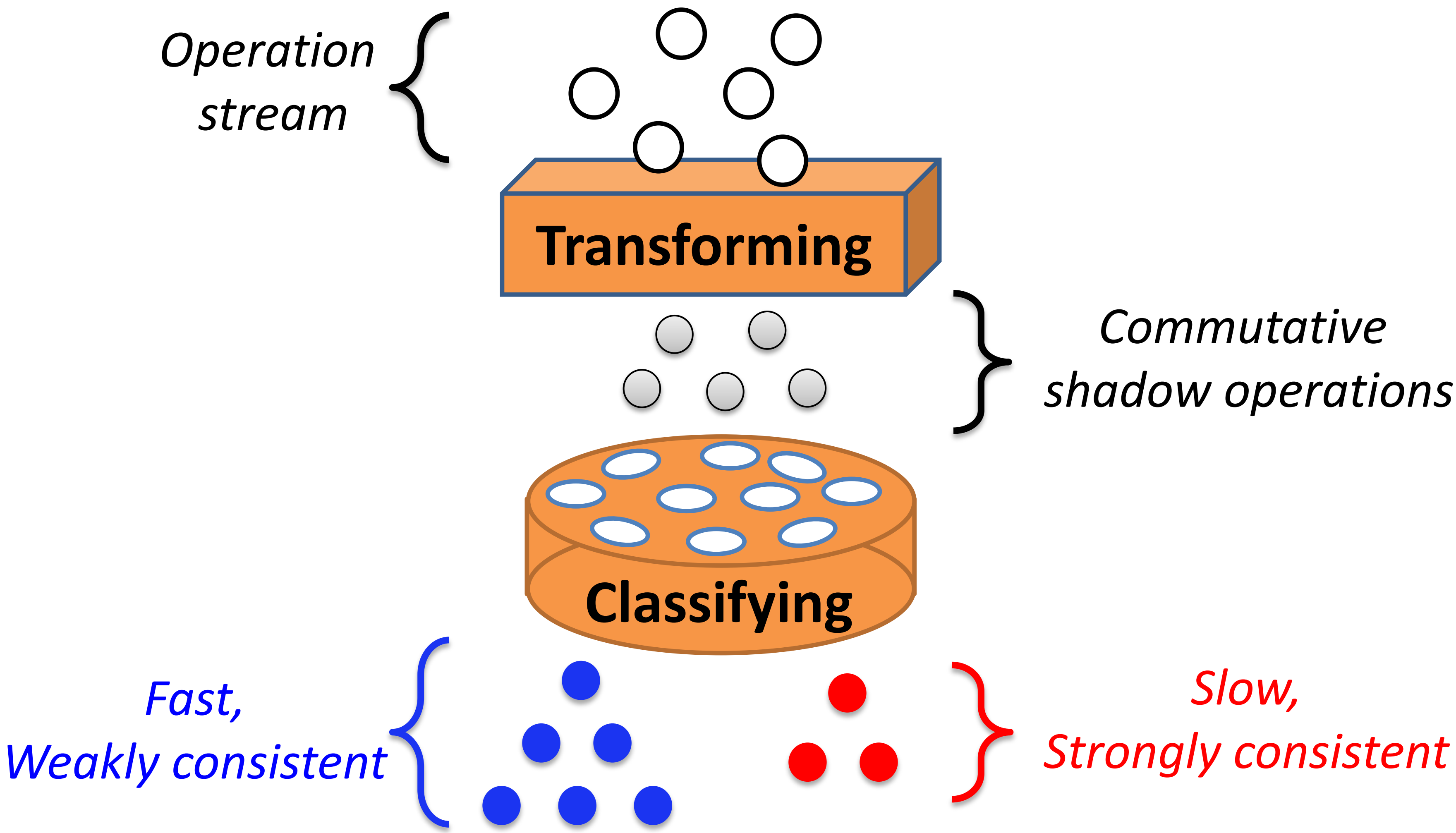
```
CREATE TABLE BankAccount(
        id INT(11) NOT NULL,
        balance INT(11) default 0,
        name char(60) default NULL,
        PRIMARY KEY (id)
) ENGINE=InnoDB
```

CRDT Annotation Example

```
@AUSET CREATE TABLE BankAccount(
                     id INT(11) NOT NULL,
 @NUMDELTA           balance INT(11) default 0,
 @LWW                name char(60) default NULL,
                     PRIMARY KEY (id)
       ) ENGINE=InnoDB
```

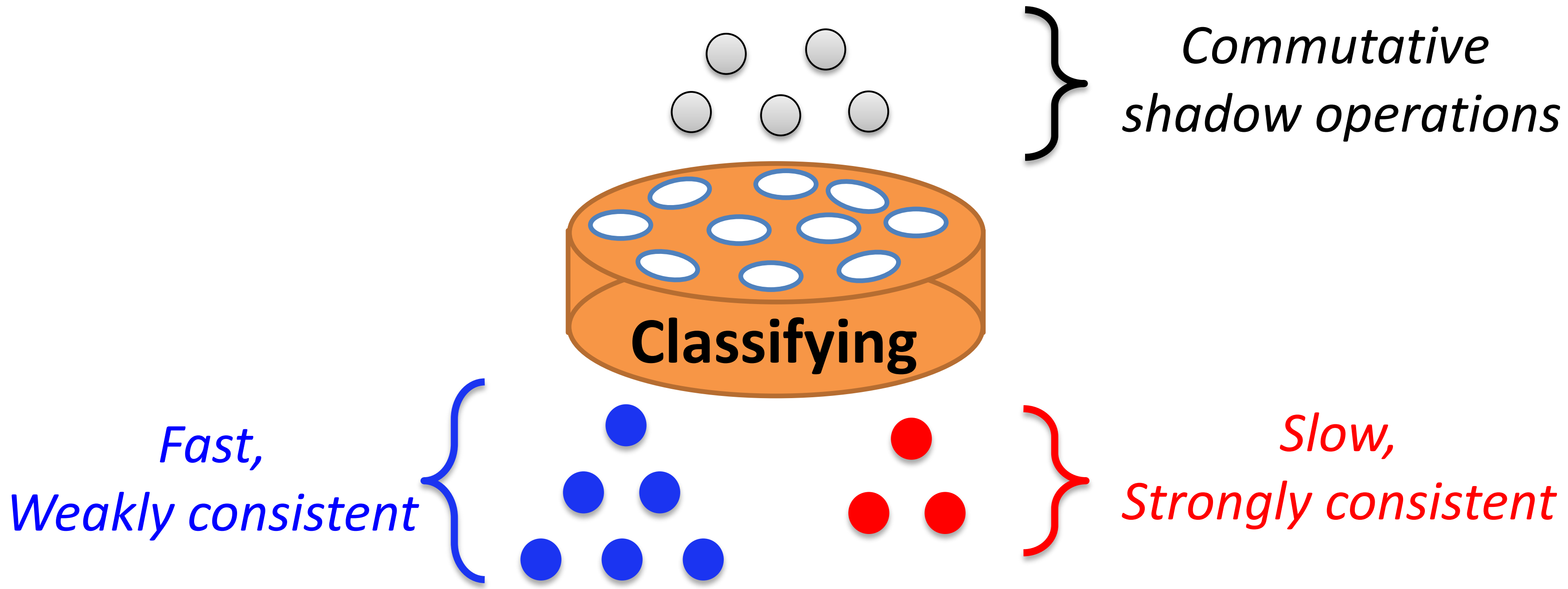
For details, refer to our paper.

SIEVE



Operation stream
Transforming
Commutative shadow operations
Classifying
Fast, Weakly consistent
Slow, Strongly consistent

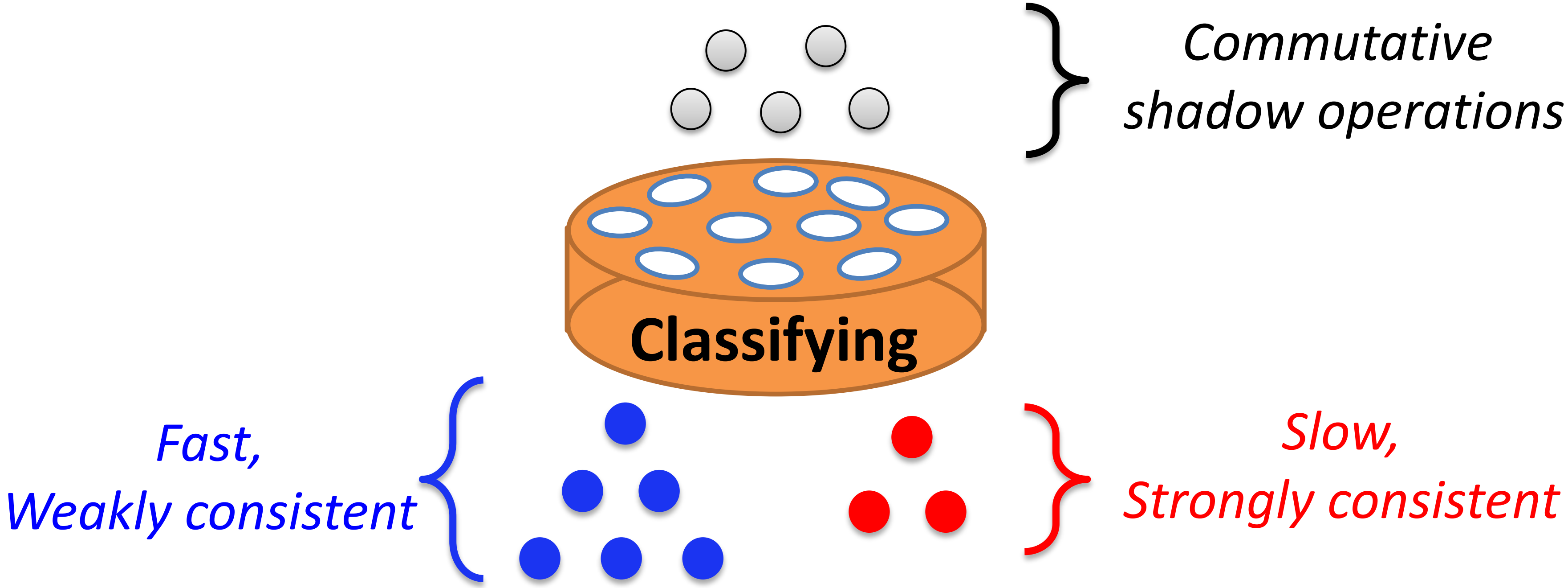
SIEVE


Commutative
shadow operations
Classifying
Fast,
Weakly consistent
Slow,
Strongly consistent

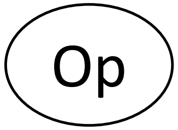
SIEVE

Challenge:

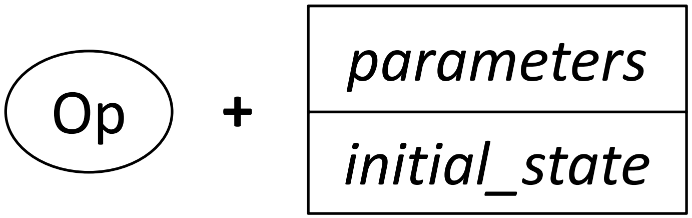
- How to classify accurately and efficiently?



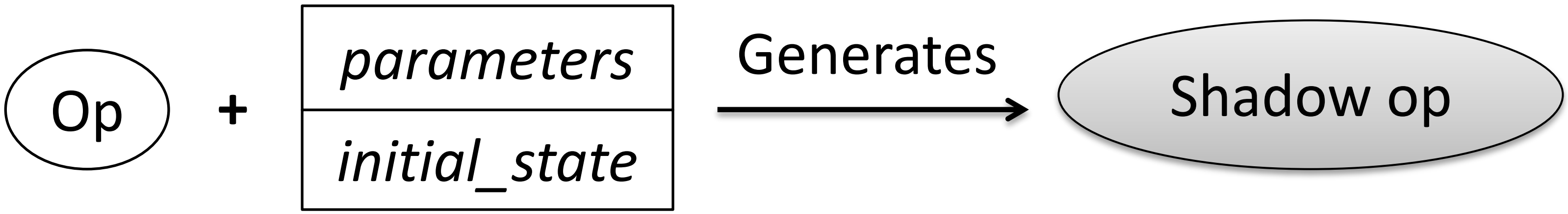
Straw man Solution



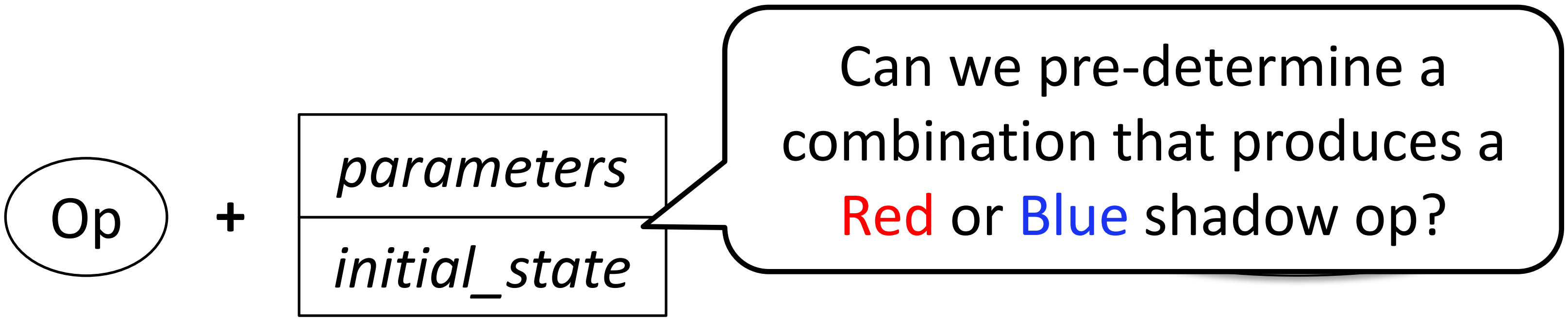
Straw man Solution



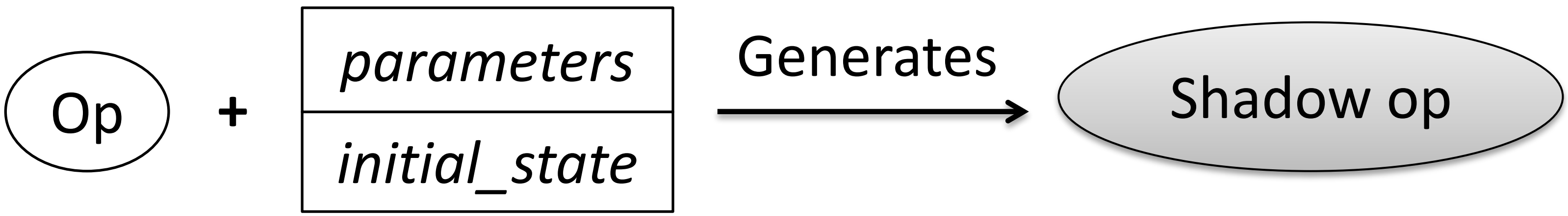
Straw man Solution



Straw man Solution



Straw man Solution



- Statically define, for each original operation, a weakest precondition (WP) for corresponding shadow op to be invariant preserving
- At runtime, we classify shadow operations by evaluating the corresponding WP

WP Computation

- Problem: WP computation is infeasible (inverting hash function)

```
Invariant: x >= 0
void foo(string s) {
  if (SHA-1(s)==SOME_CONSTANT) {
     if (x>=10)
        x -= 10;
  } else
       x +=10;
}
```

WP Computation

- Problem: WP computation is infeasible (inverting hash function)
  - Classification will be conservative.

```
Invariant: x >= 0
void foo(string s) {
  if (SHA-1( ==S    E_CONSTANT) {
      ;(
  } else
        x +=10
}
```

Weakest precondition must be FALSE!

WP Computation

- Problem: WP computation is infeasible (inverting hash function)
  - Classification will be conservative.
- Observation: side effects are simple.

Invariant: x >= 0

```
void foo(string s) {
  if (SHA-1(    ==S    E_CONSTANT) {
  } else
      x +=10
}
```

Weakest precondition must be FALSE!

WP Computation

- Problem: WP computation is infeasible (inverting hash function)
  - Classification will be conservative.
- Observation: side effects are simple.

Invariant: x >= 0

```
void foo(string s) {
  if (SHA-1(s)==SOME_CONSTANT) {
    if (x>=10)
      x -= 10;
  } else
    x +=10;
}
```

Apply -10 to x

Do nothing

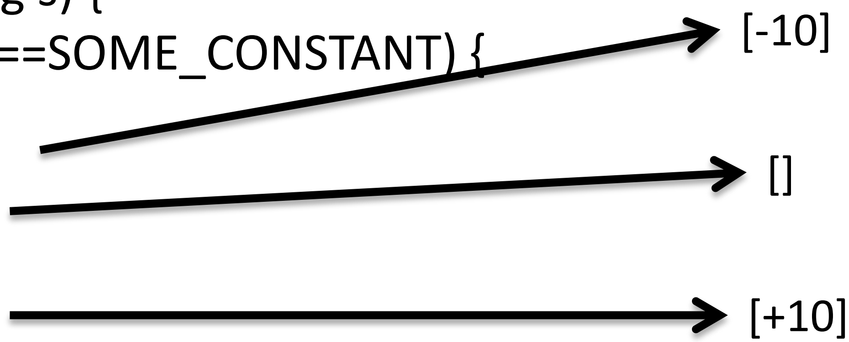
Apply 10 to x

Path-basis Analysis

- Creates a ***template*** per control flow path to capture all possible shadow operations following that path

Invariant: x >= 0

```
void foo(string s) {
  if (SHA-1(s)==SOME_CONSTANT) {
     if (x>=10)
        x -= 10;
  } else
       x +=10;
}
```

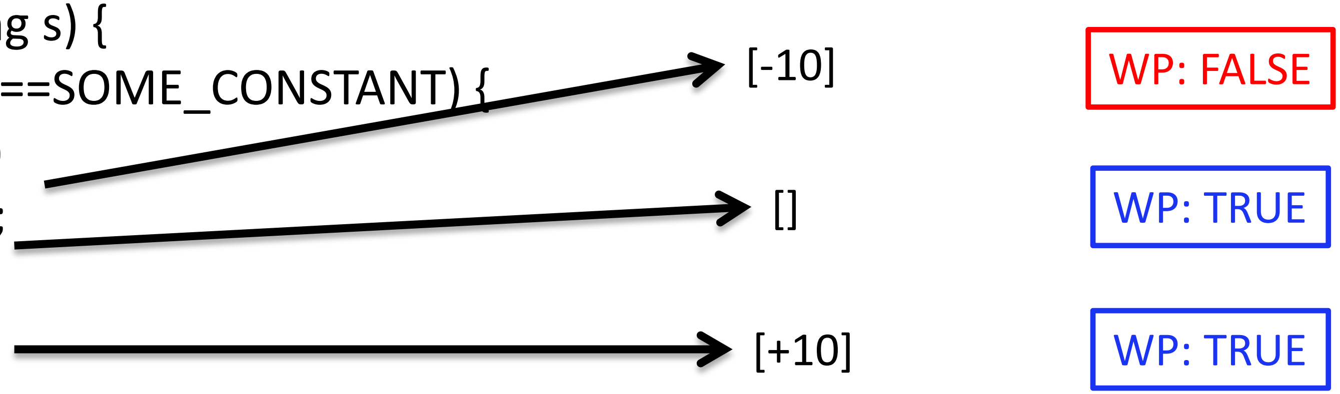


Path-basis Analysis

- Creates a ***template*** per control flow path to capture all possible shadow operations following that path
- WP computation is done over parameters of CRDT invocations in templates

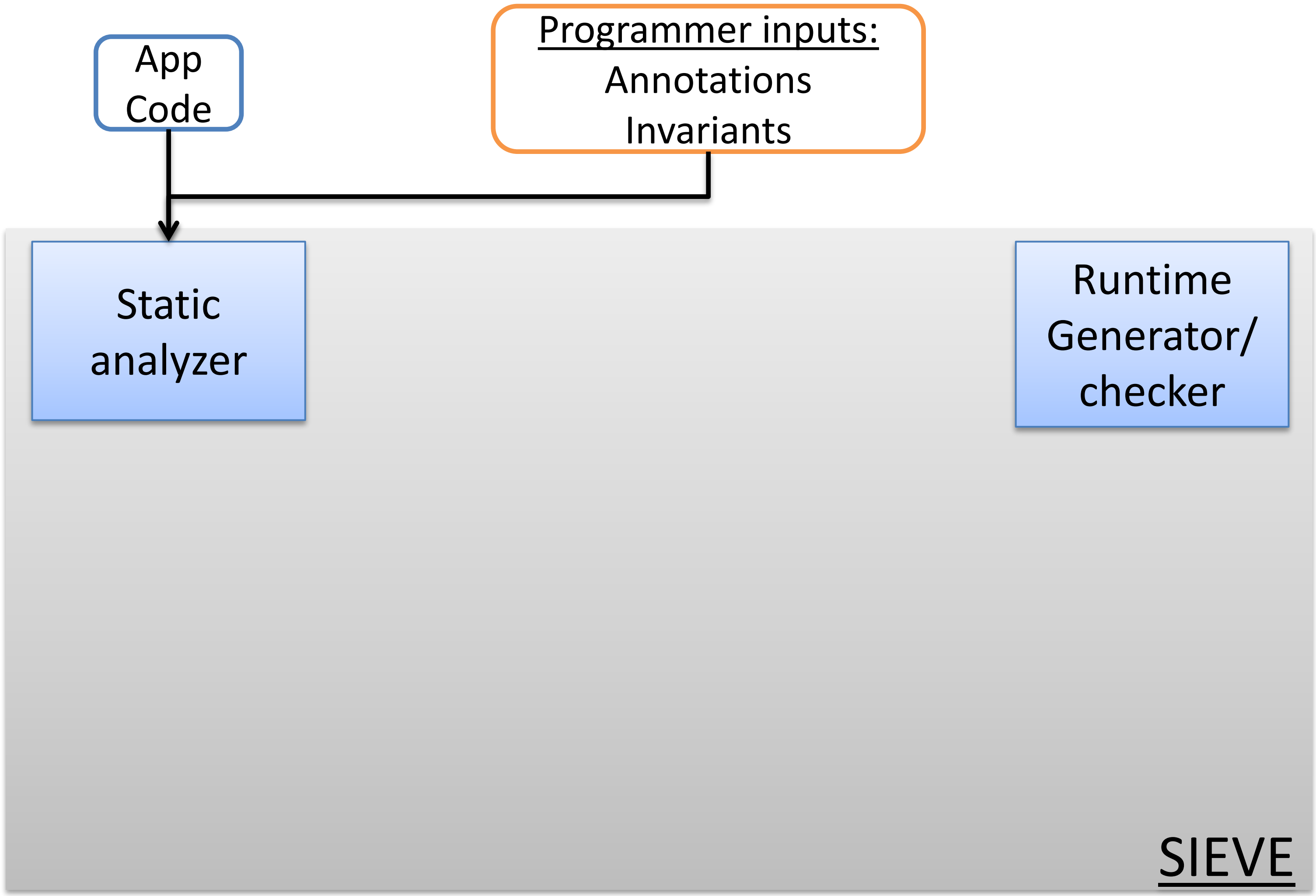
Invariant: x >= 0

```
void foo(string s) {
  if (SHA-1(s)==SOME_CONSTANT) {
    if (x>=10)
      x -= 10;
  } else
    x +=10;
}
```

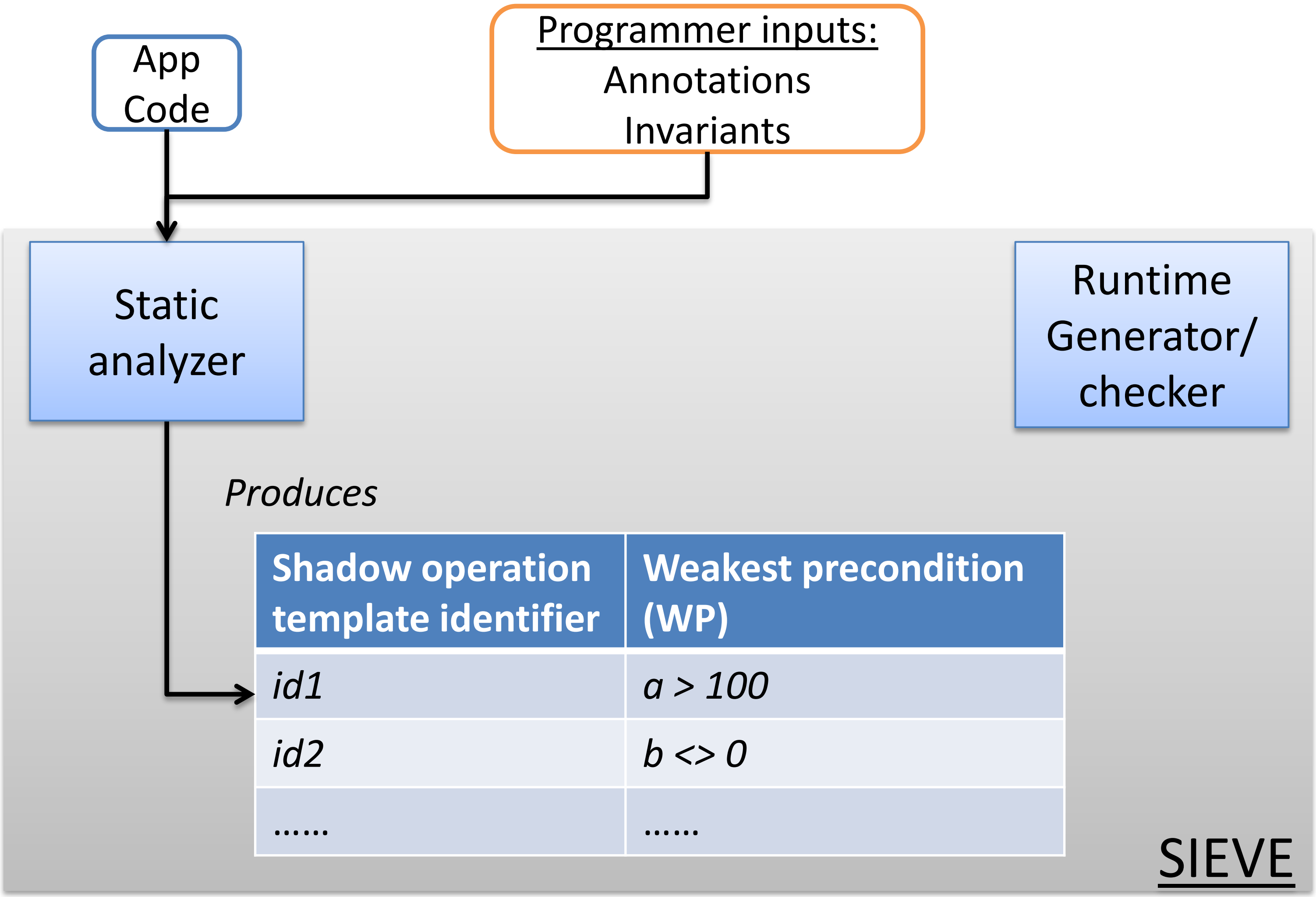


Do most (not all) work offline!

Do most (not all) work offline!

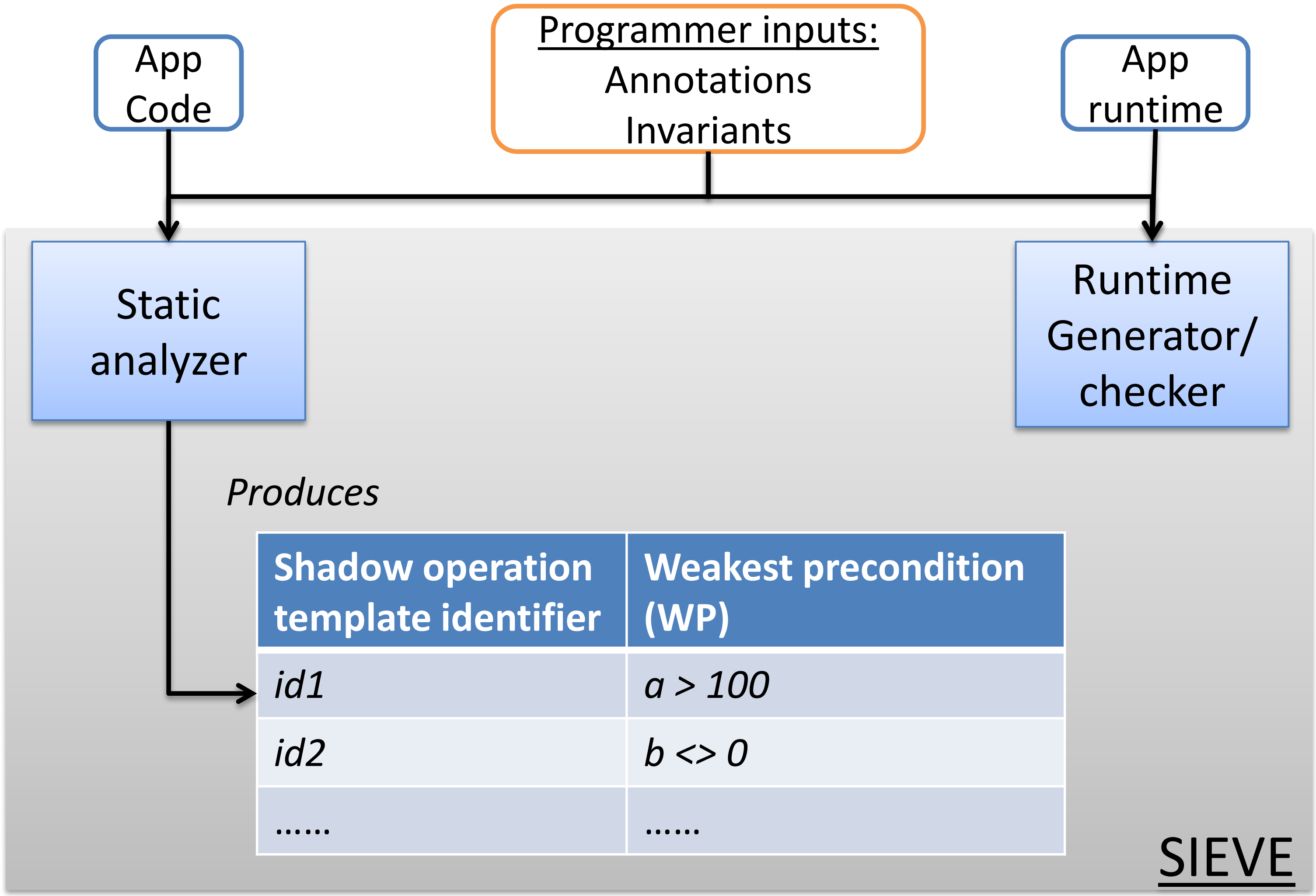


Do most (not all) work offline!



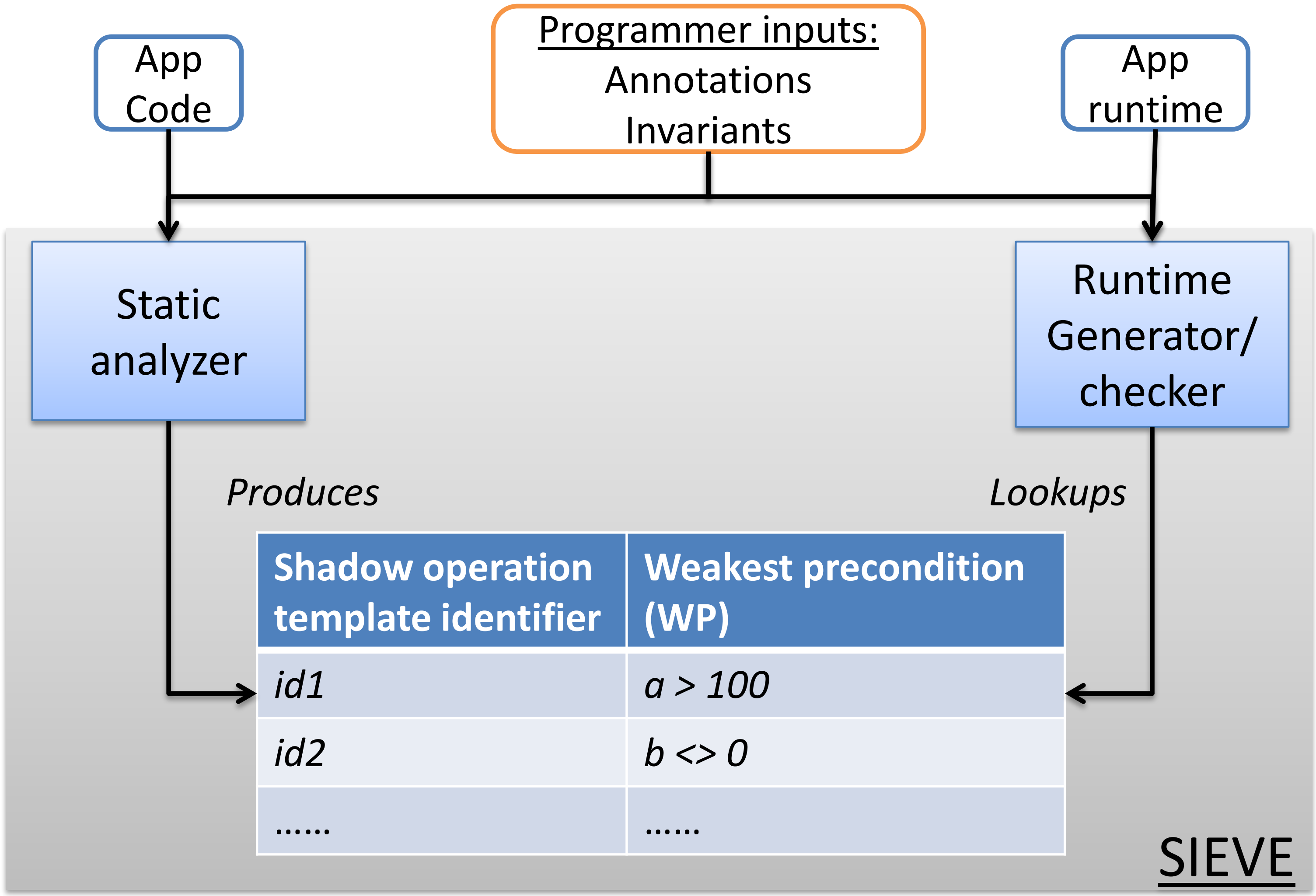
Shadow operation template identifier	Weakest precondition (WP)
id1	*a > 100*
id2	*b <> 0*
……	……

Do most (not all) work offline!



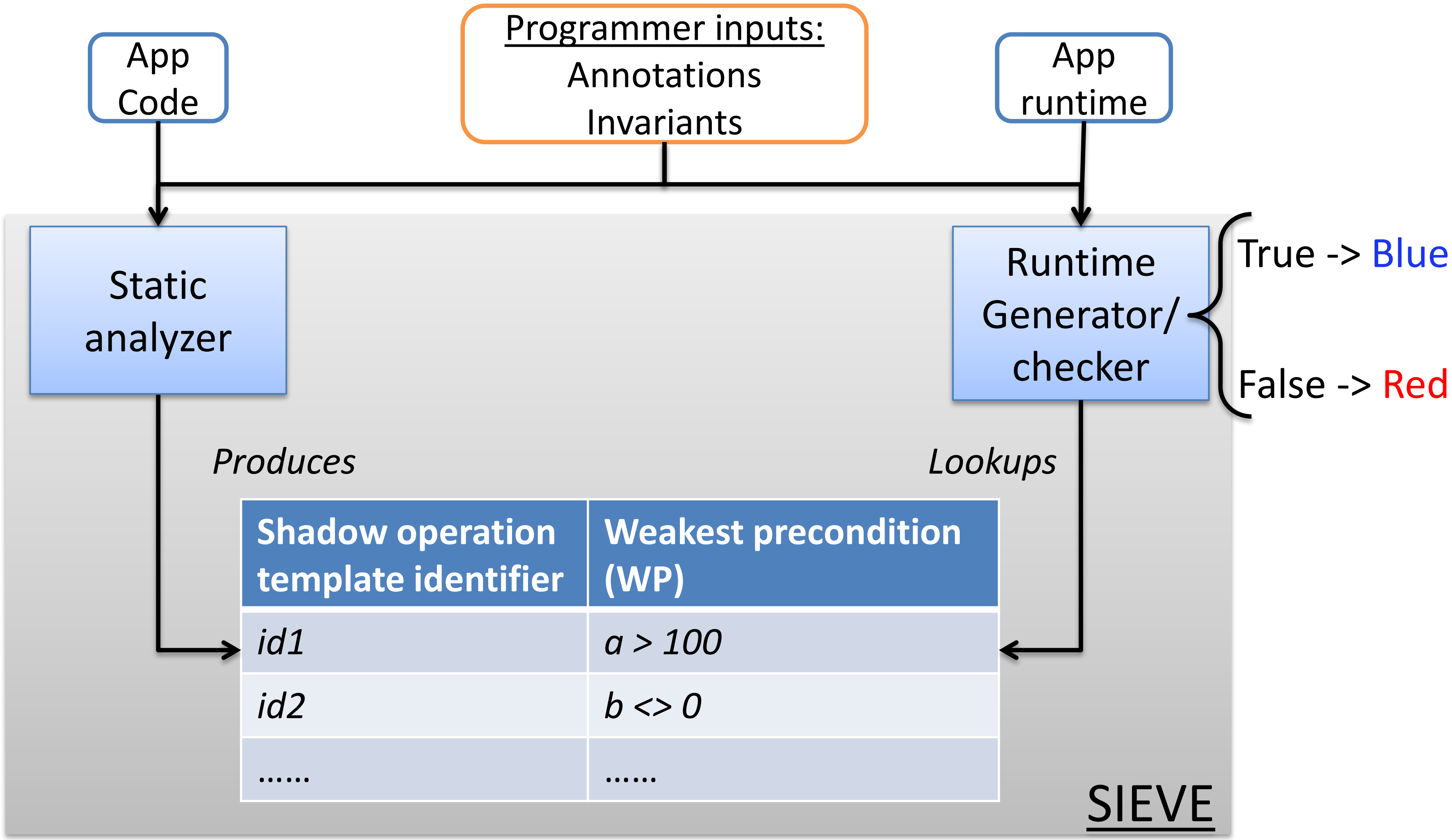
Shadow operation template identifier	Weakest precondition (WP)
id1	*a > 100*
id2	*b <> 0*
……	……

Do most (not all) work offline!



Shadow operation template identifier	Weakest precondition (WP)
id1	*a > 100*
id2	*b <> 0*
……	……

Do most (not all) work offline!



Shadow operation template identifier	Weakest precondition (WP)
id1	*a > 100*
id2	*b <> 0*
……	……

Evaluation

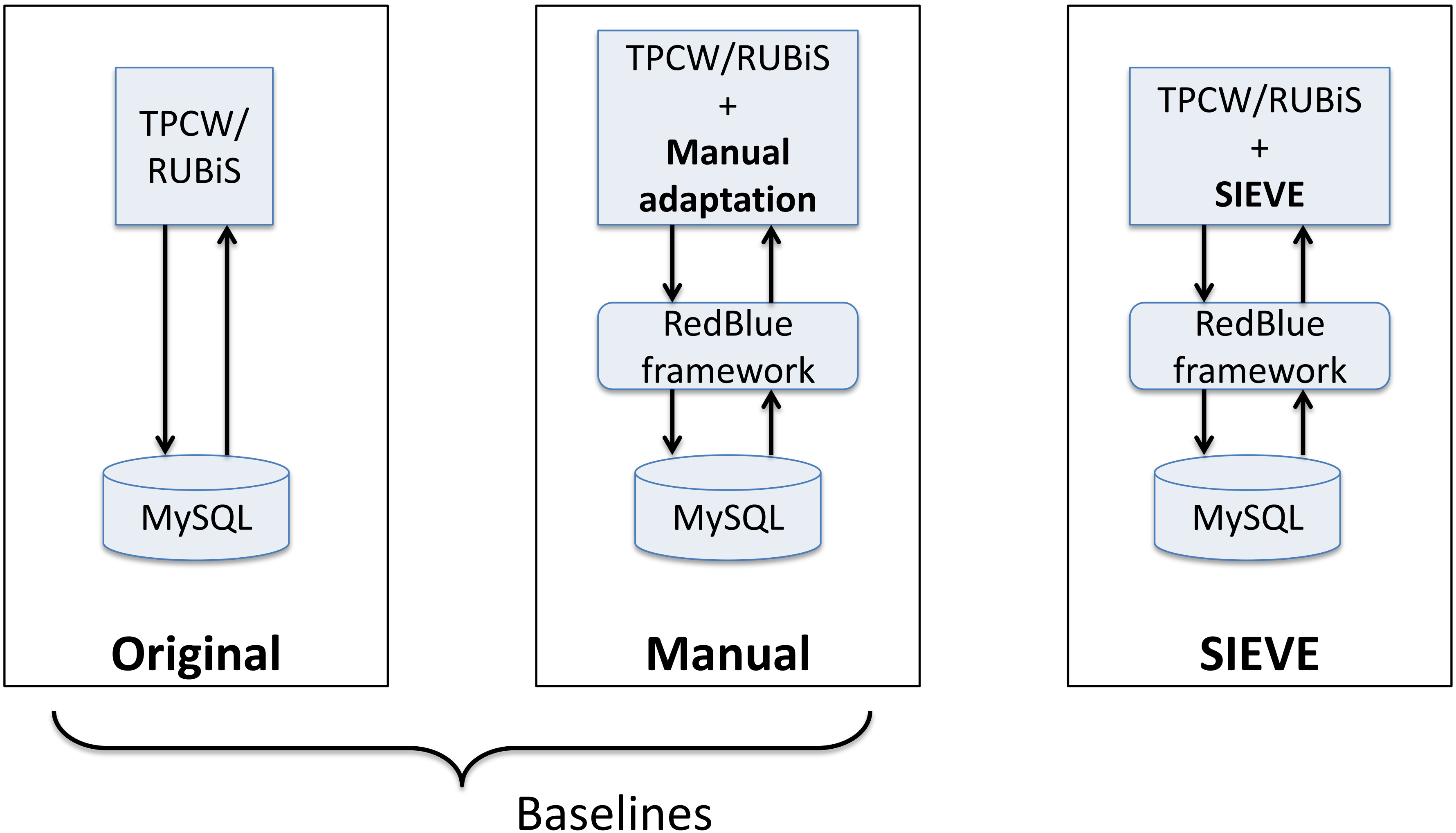
Questions

- Application adaptation
  - How easy is it to adapt apps to SIEVE?
- Static analysis
  - How long does the static analysis process take?
  - How well does the static analysis scale?
- Runtime part
  - Is the runtime classification accurate?
  - What is the overhead?
  - How does the replicated application perform?

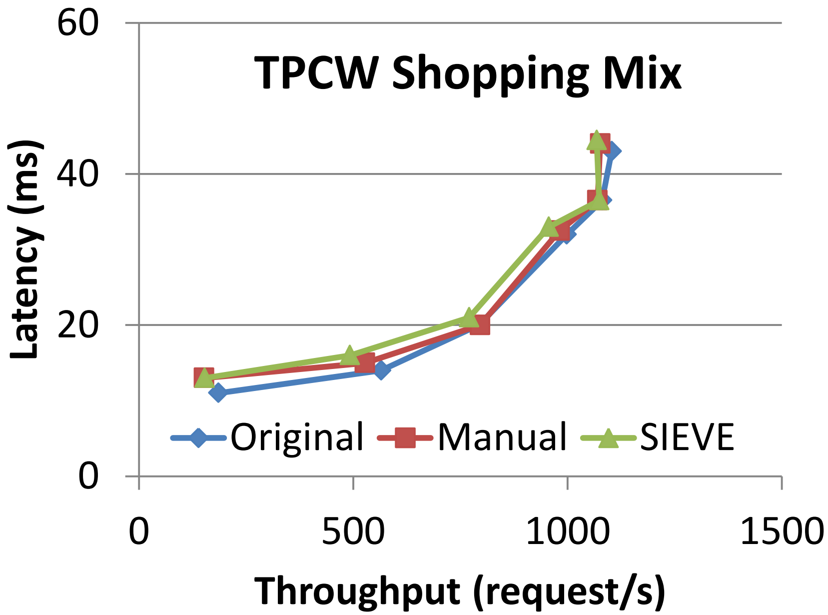
Questions

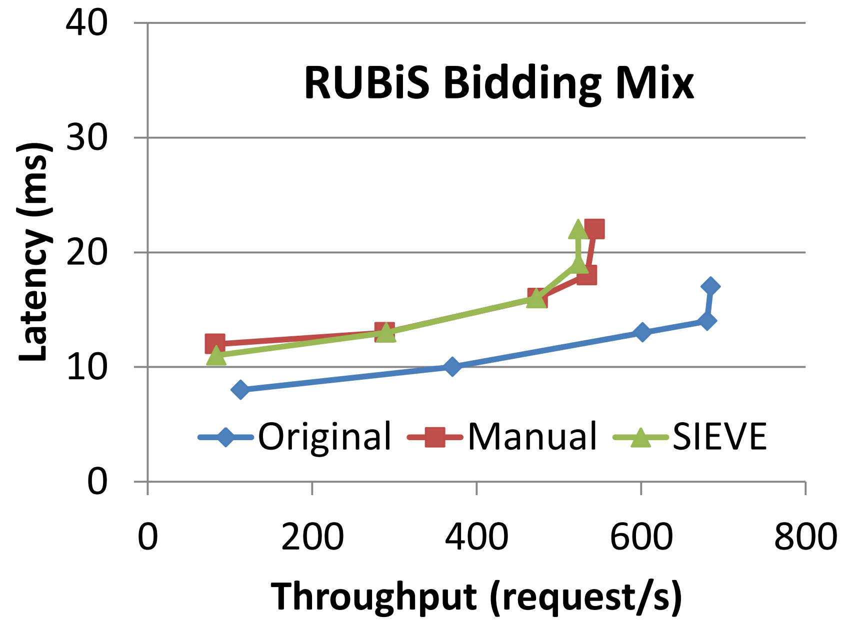
- Application adaptation
  - How easy is it to adapt apps to SIEVE?
- Static analysis
  - How long does the static analysis process take?
  - How well does the static analysis scale?
- Runtime part
  - Is the runtime classification accurate?
  - What is the overhead?
  - How does the replicated application perform?

Experimental Setup



Runtime Overhead





- SIEVE performs almost as well as manual adaptation
- Runtime labeling takes negligible time
  - TPCW : 0.064 ± 0.002 ms
  - RUBiS : 0.072 ± 0.001 ms

Conclusion

SIEVE *automatically* and *efficiently* chooses weak consistency (blue) *whenever possible*, and strong consistency (red) *when needed*, only requiring little programmer input.

Automating the Choice of Consistency Levels in Replicated Systems

Thanks for your attention!

Q and A!