



NOVALINCS

# MAINTAINING SQL INVARIANTS IN WEAKLY CONSISTENT DATABASES

Nuno Preguiça (NOVA LINCS, FCT/Universidade NOVA de Lisboa)

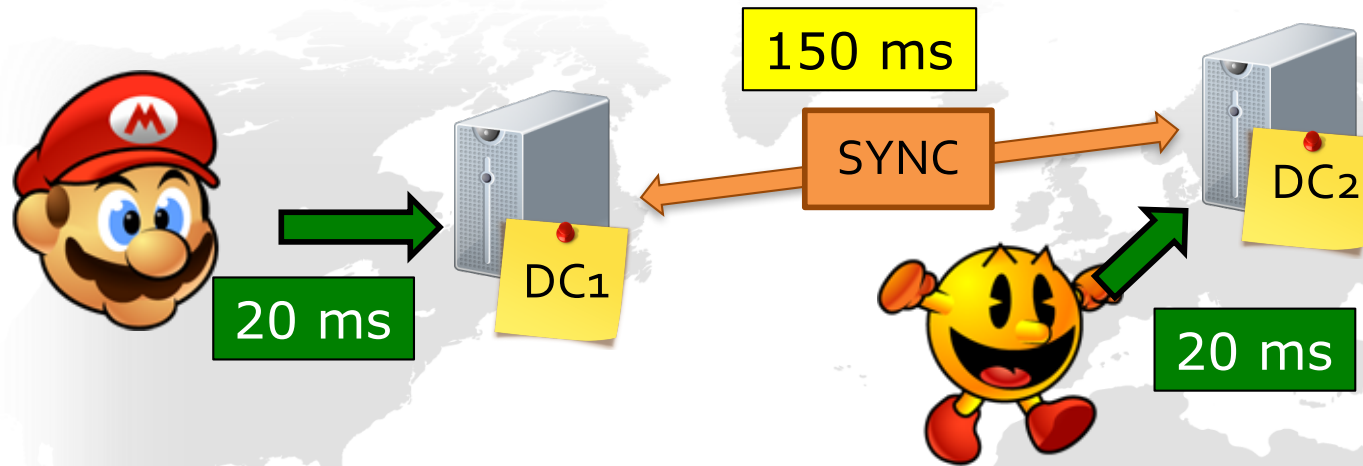
Joint work with:

Valter Balegas, Cheng Li (MPI, now Oracle), João Sousa, David Lopes, Sérgio Duarte, Carla Ferreira, João Leitão, Allen Clement (MPI, now Google), Viktor Vafeiadis (MPI), Rodrigo Rodrigues (now Inesc-Id/IST)

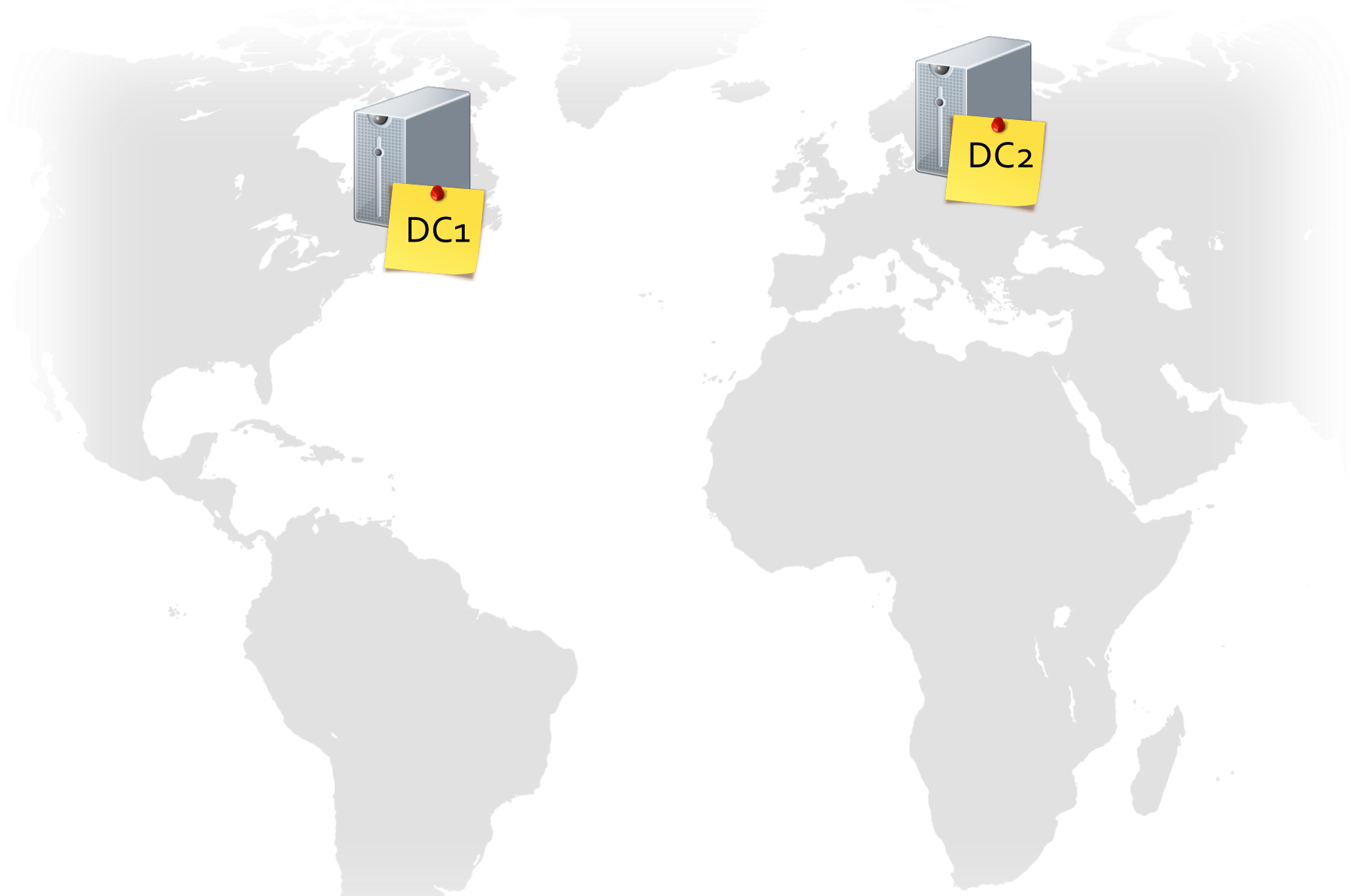
# INTERNET SERVICES NOWADAYS

- Services operate on a global scale.
- An unprecedented number of people are using Internet services.
- Systems use geo-replication for low latency and high availability.

# GEO-REPLICATION



# GEO-REPLICATION



# GEO-REPLICATION



```

create table player( id varchar(20), primary key id)
create table tournament( id varchar(20), primary key id)
create table pt( p varchar(20), t varchar(20), foreign key (p) REFERENCES
player (id), foreign key (t) REFERENCES tournament (id))
  
```

Player
Sonic
Pacman
Mario

PT
Sonic, A
Sonic, B

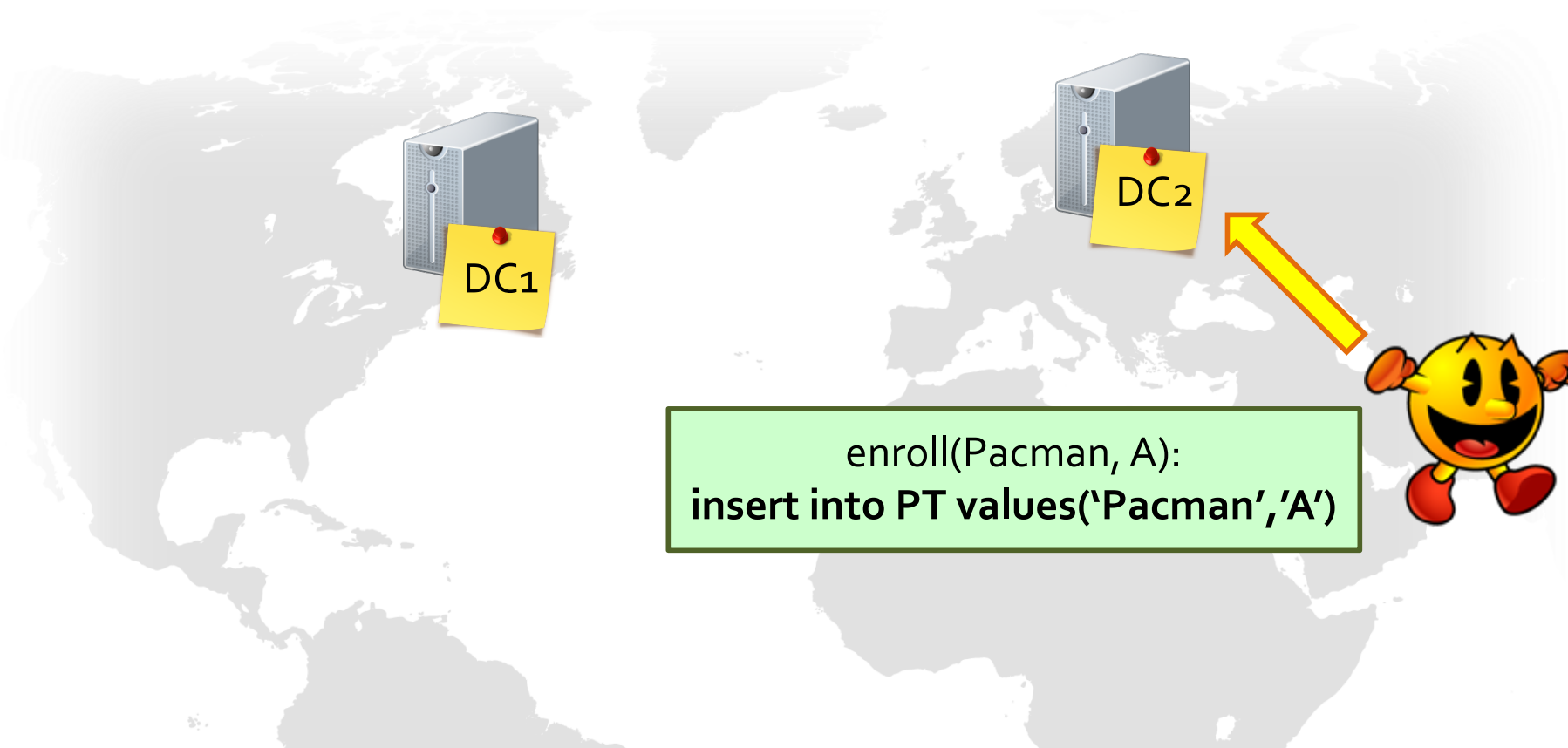
Tournament
A
B

Player
Sonic
Pacman
Mario

PT
Sonic, A
Sonic, B

Tournament
A
B

# GEO-REPLICATION



Player	PT	Tournament
Sonic	Sonic, A	A
Pacman	Sonic, B	B
Mario		

Player	PT	Tournament
Sonic	Sonic, A	A
Pacman	Sonic, B	B
Mario	Pacman, A	

# GEO-REPLICATION

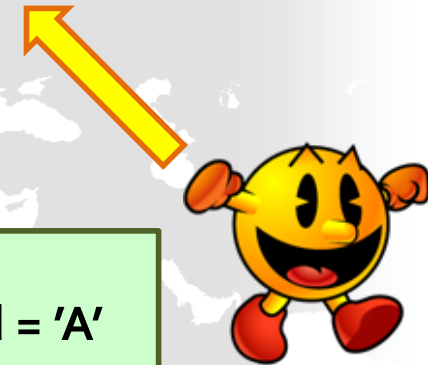


```
enroll(Mario, A):
insert into PT values('Mario','A')
```

Player	PT	Tournament
Sonic	Sonic, A	A
Pacman	Sonic, B	B
Mario	Mario, A	

Player	PT	Tournament
Sonic	Sonic, A	A
Pacman	Sonic, B	B
Mario	Pacman, A	

# GEO-REPLICATION



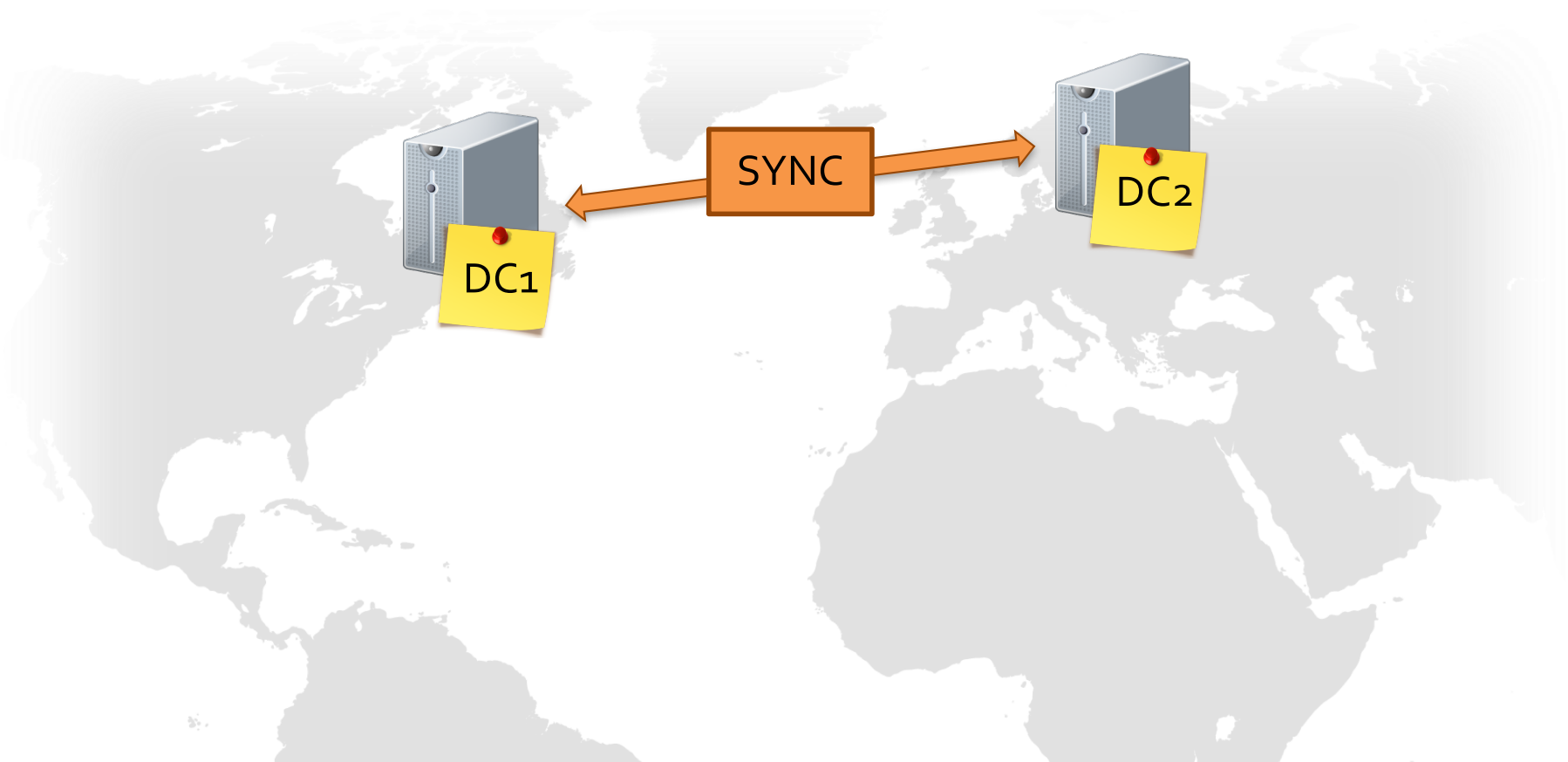
removeTournament(A):  
 delete from tournament where id = 'A'  
 delete from PT where t = 'A'

Player	PT	Tournament
Sonic	Sonic, A	A
Pacman	Sonic, B	B
Mario	Mario, A	

Player	PT	Tournament
Sonic	<del>Sonic, A</del>	<del>A</del>
Pacman	Sonic, B	B
Mario	<del>Pacman, A</del>	



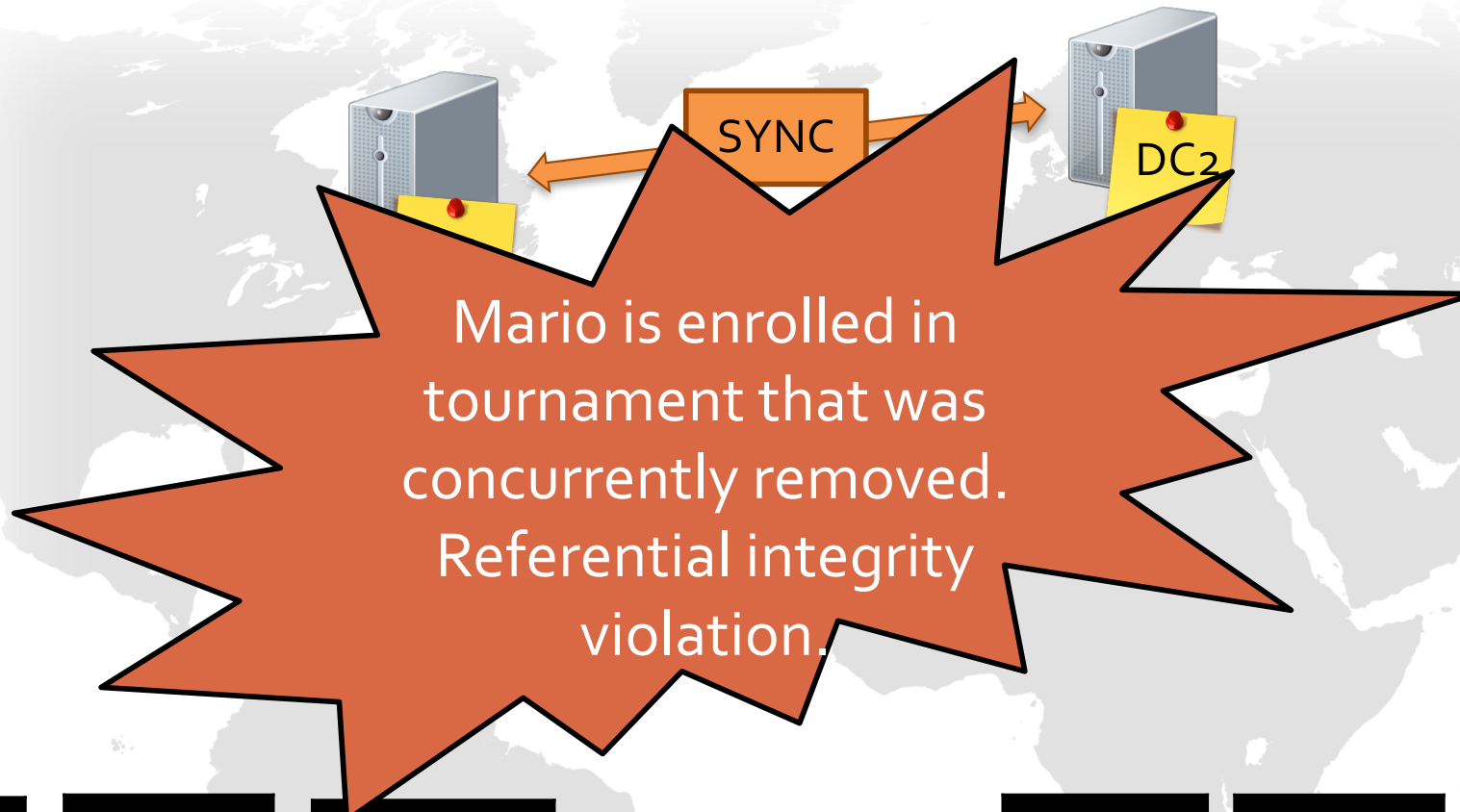
# GEO-REPLICATION



Player	PT	Tournament
Sonic	Sonic, A	A
Pacman	Sonic, B	B
Mario	Mario, A	

Player	PT	Tournament
Sonic	<del>Sonic, A</del>	<del>A</del>
Pacman	Sonic, B	B
Mario	<del>Pacman, A</del>	

# GEO-REPLICATION



Player	PT	Tournament
Sonic	<del>Sonic, A</del>	<del>A</del>
Pacman	Sonic, B	B
Mario	<del>Pacman, A</del>	
	Mario, A	

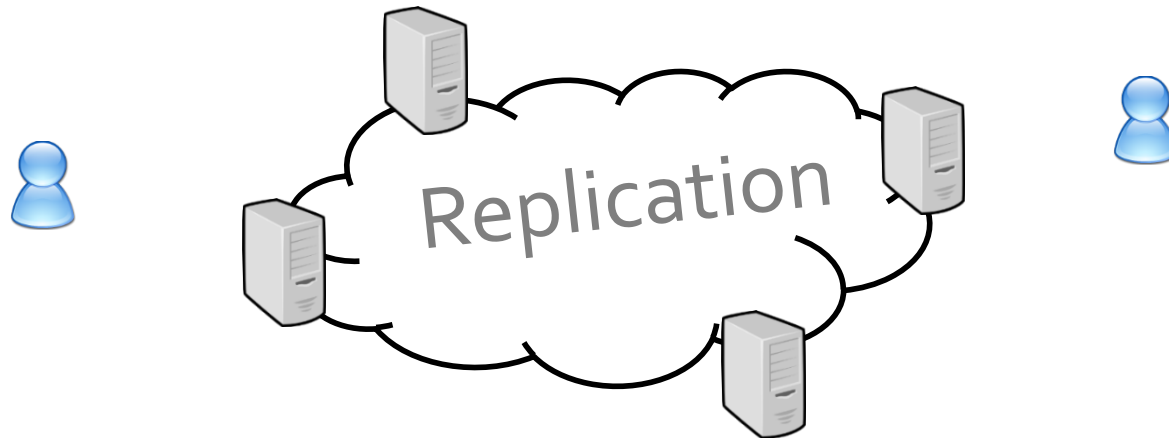
Player	PT	Tournament
Sonic	<del>Sonic, A</del>	<del>A</del>
Pacman	Sonic, B	B
Mario	<del>Pacman, A</del>	
	Mario, A	

# OUTLINE

- Context / problem
- First take: Sieve
- Second take: SQL IPA
- Final remarks

# RedBlue Consistency

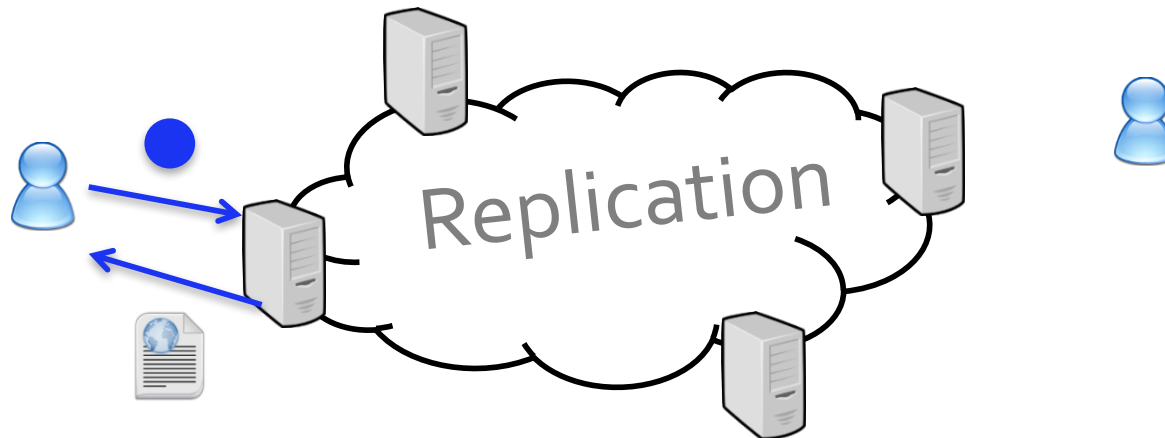
Builds replicated systems that are fast



# RedBlue Consistency

Builds replicated systems that are fast

Blue ops: local, fast, weakly consistent



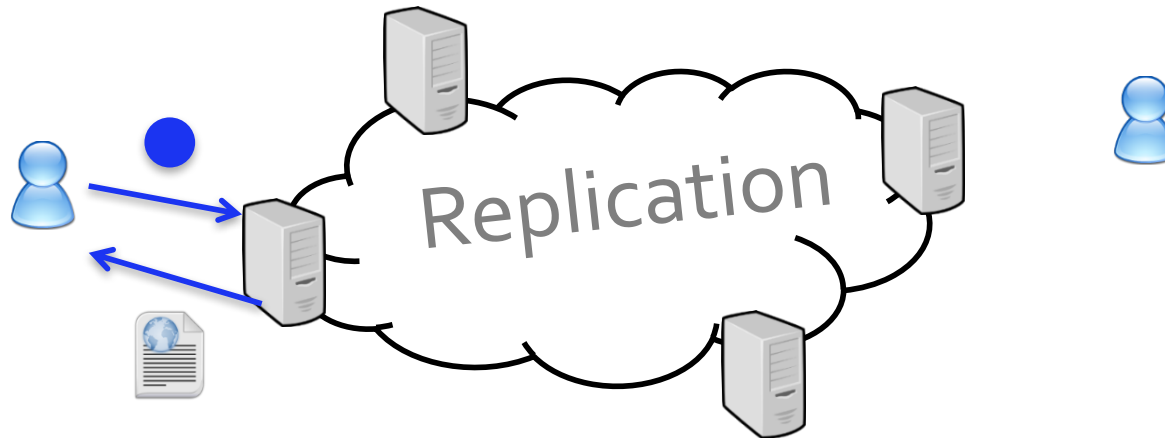
# RedBlue Consistency

Builds replicated systems that are fast and correct

Blue ops: local, fast, weakly consistent

State  
convergence

Invariant  
preservation



# RedBlue Consistency

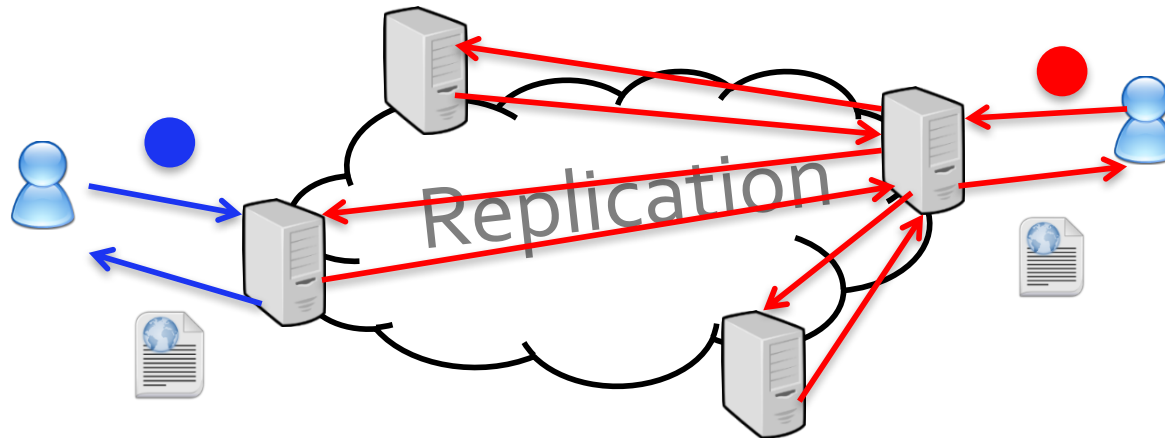
Builds replicated systems that are fast and correct

Blue ops: local, fast, weakly consistent

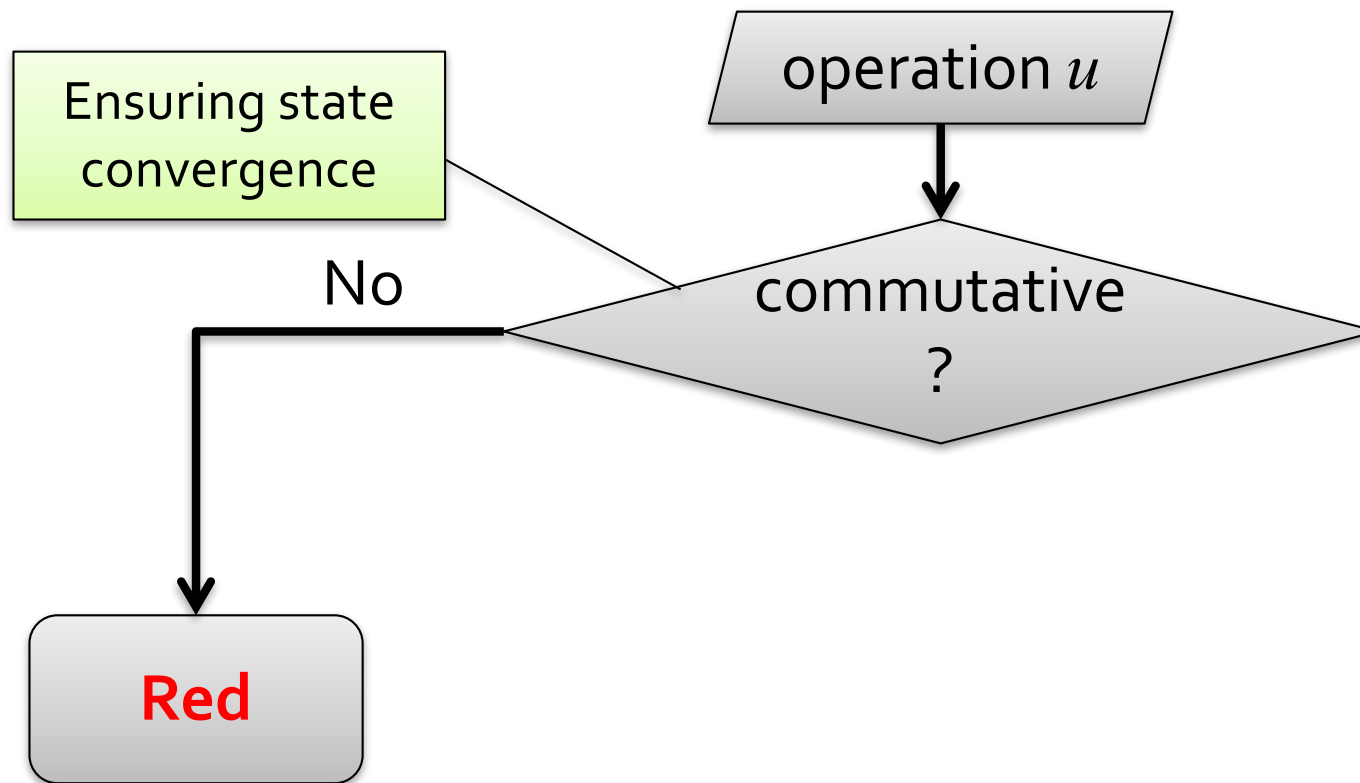
Red ops: global, slow, strongly consistent

State convergence

Invariant preservation

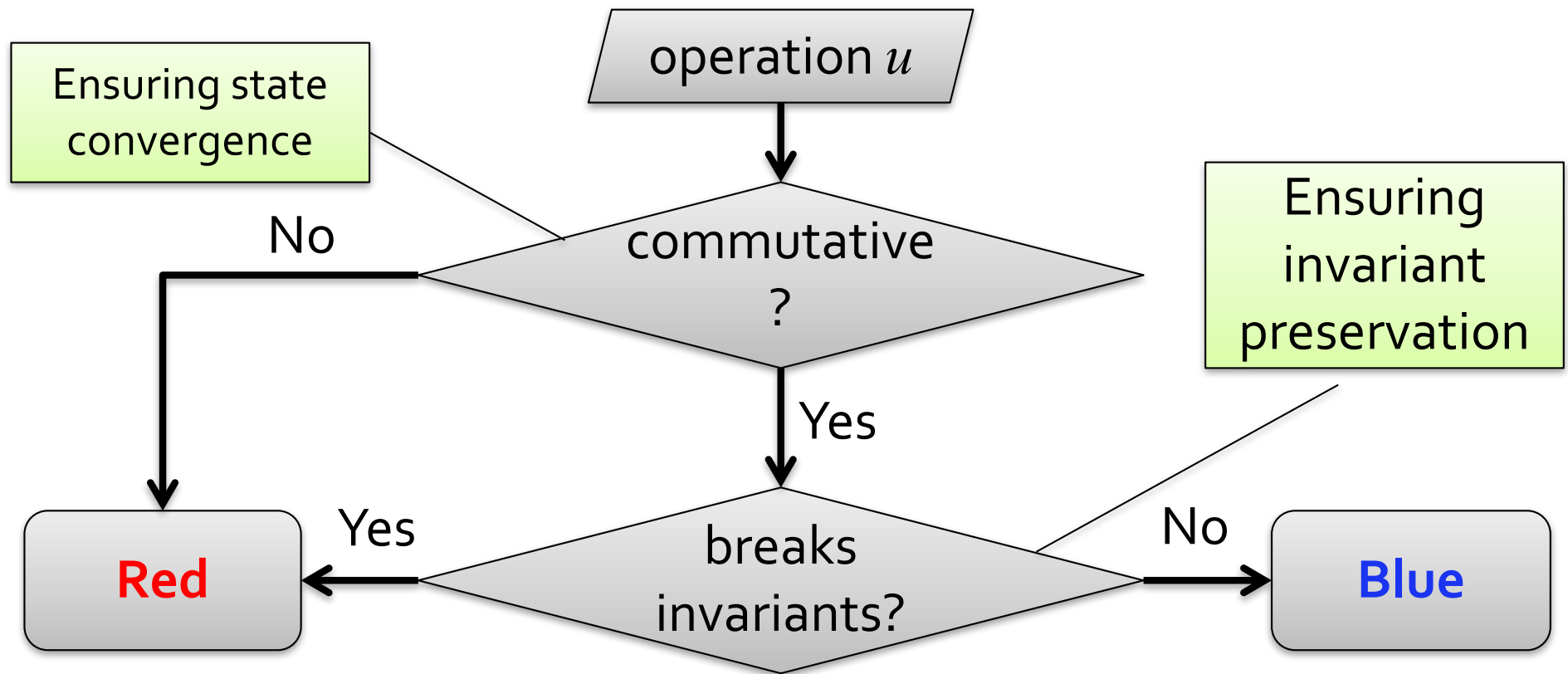


# Choosing between **Blue** or **Red**



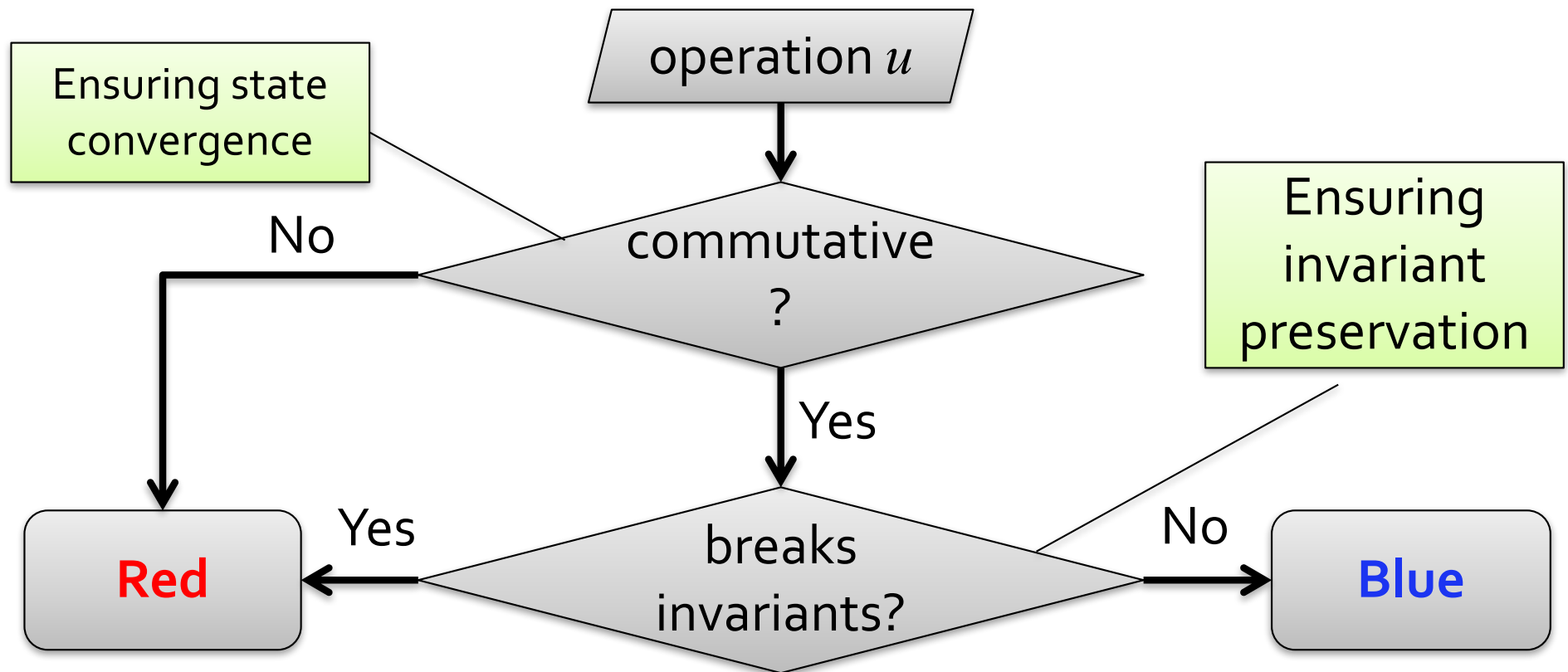


# Choosing between **Blue** or **Red**

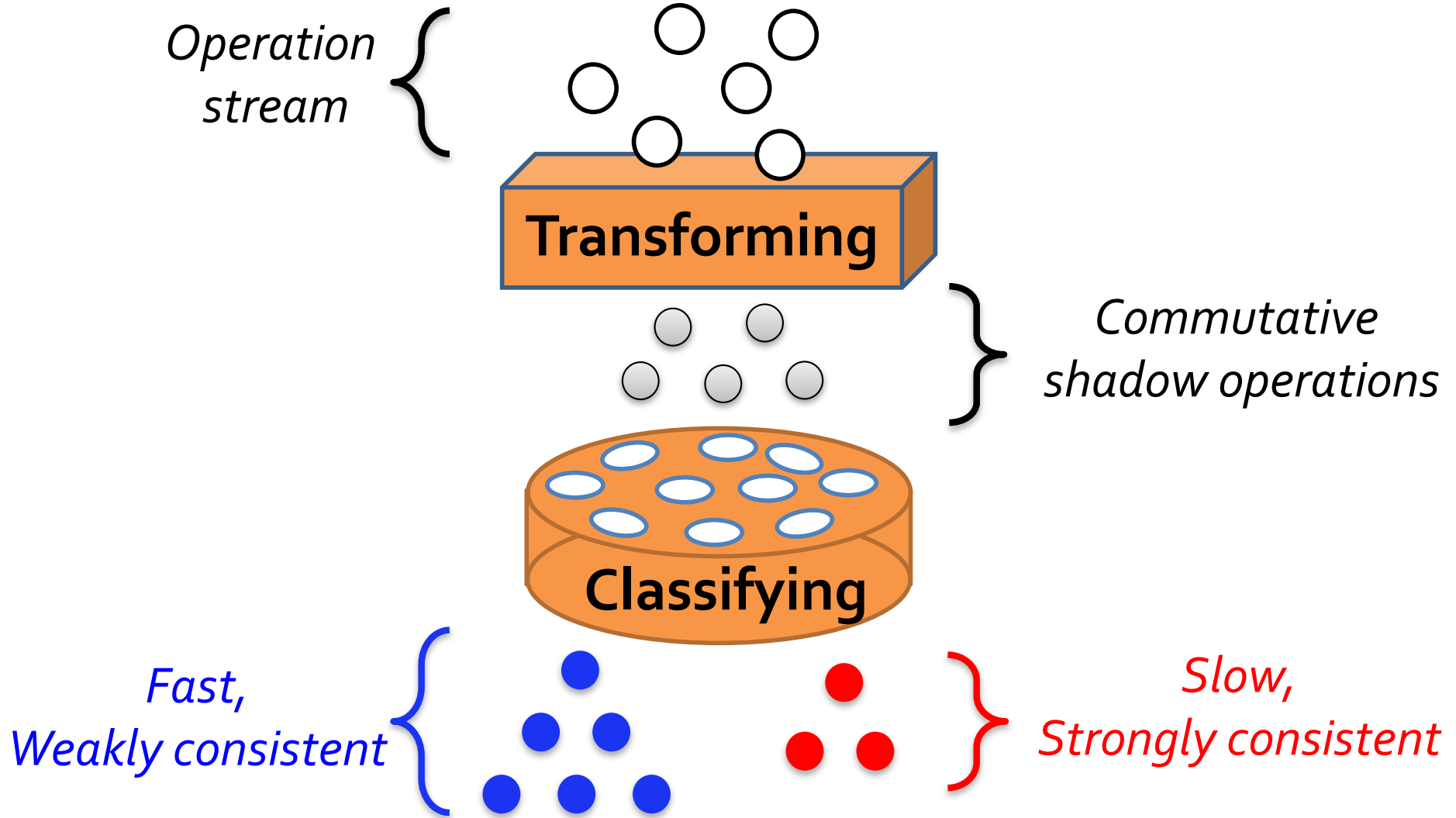


# Choosing between **Blue** or **Red**

Good performance obtained if **blue** ops dominate op space

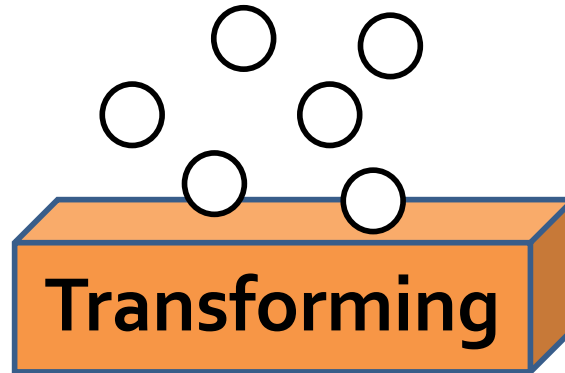


# SIEVE



# SIEVE

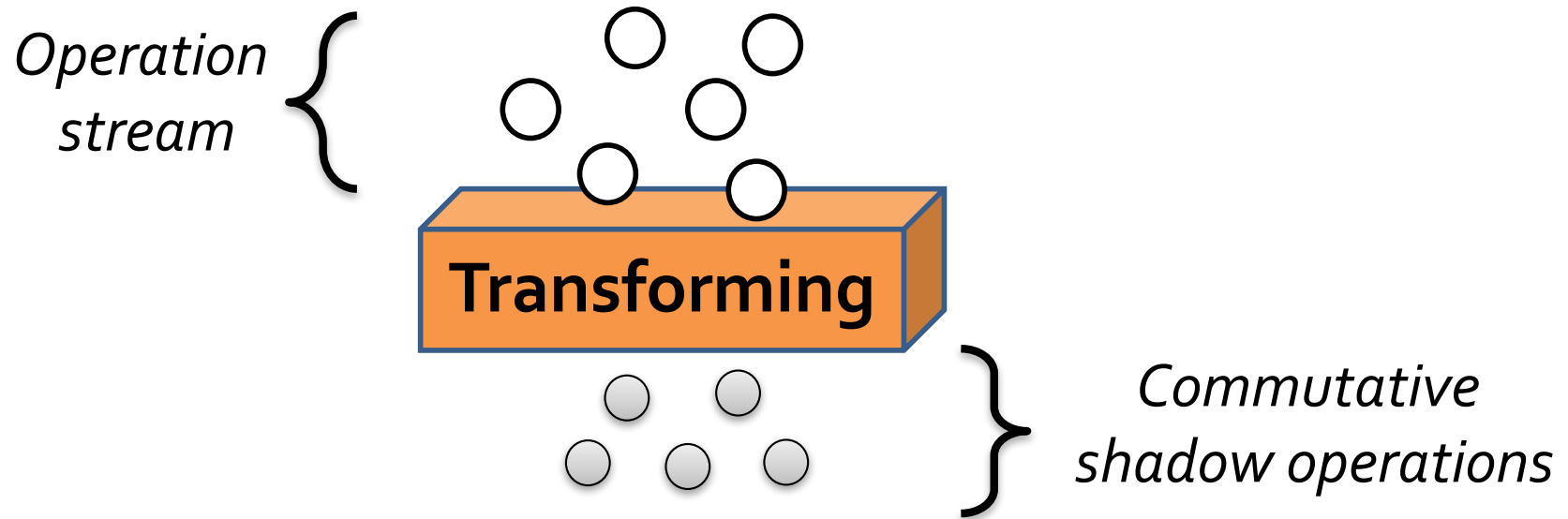
*Operation stream* {



} *Commutative shadow operations*



# SIEVE

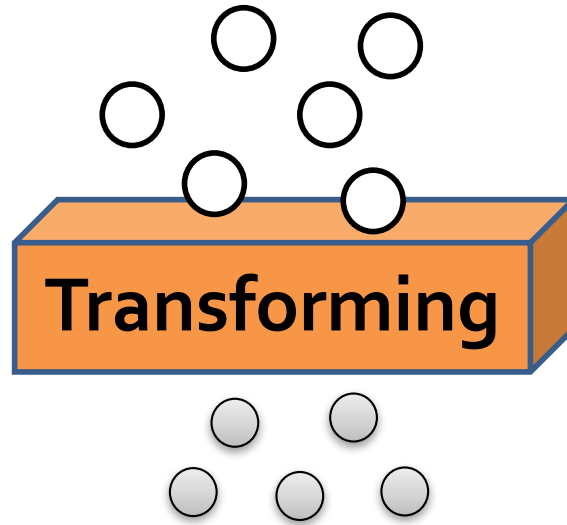


## Challenges:

- Making arbitrary side effects commute
- Minimizing human intervention

# SIEVE

Operation stream



Challenges:

- Making operations commute
- Avoiding manual intervention

**Approach: model the database using commutative replicated data types (CRDTs)**

# CRDT Annotation Example

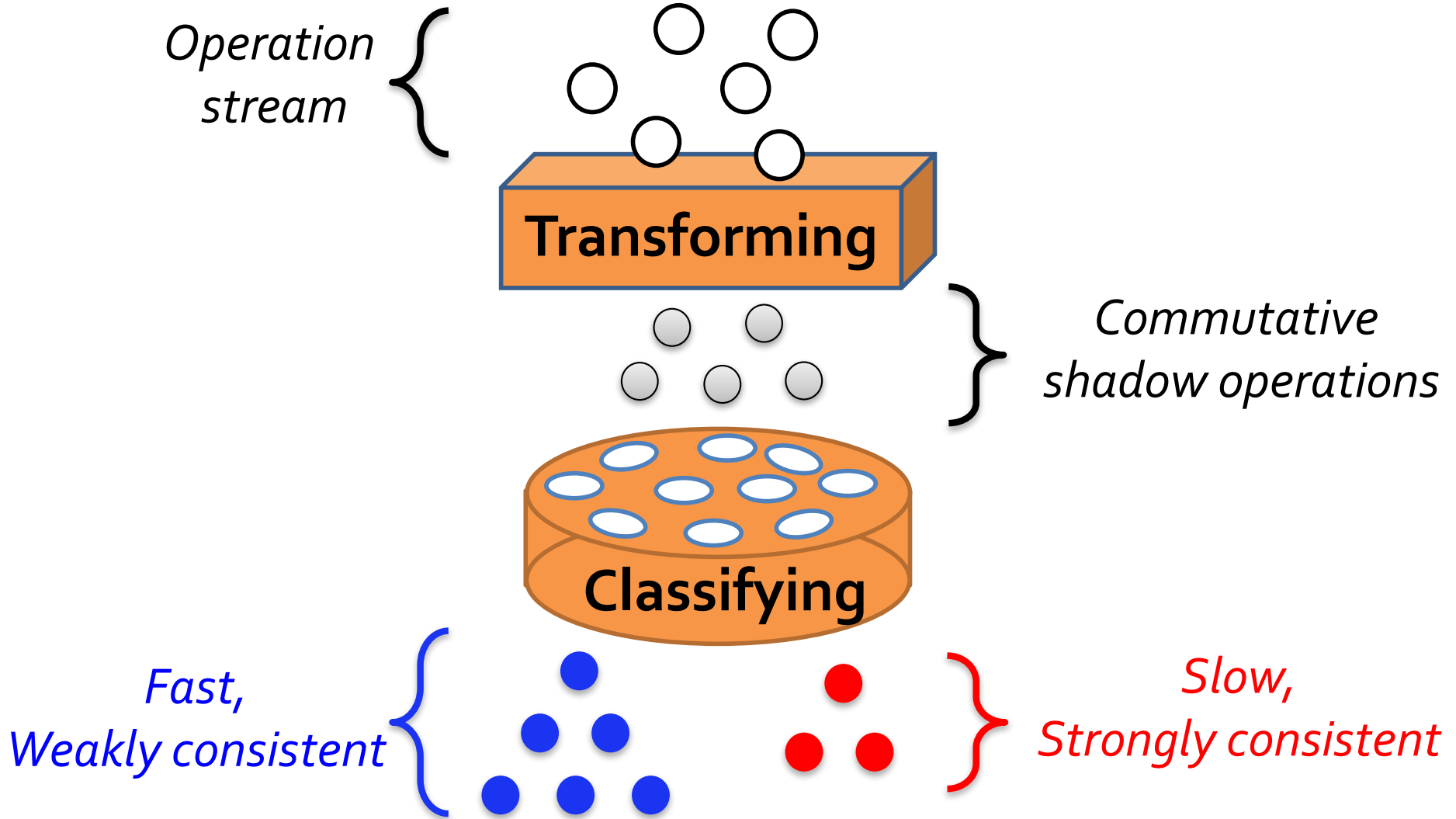
```
CREATE TABLE BankAccount(  
    id INT(11) NOT NULL,  
    balance INT(11) default 0,  
    name char(60) default NULL,  
    PRIMARY KEY (id)  
) ENGINE=InnoDB
```

# CRDT Annotation Example

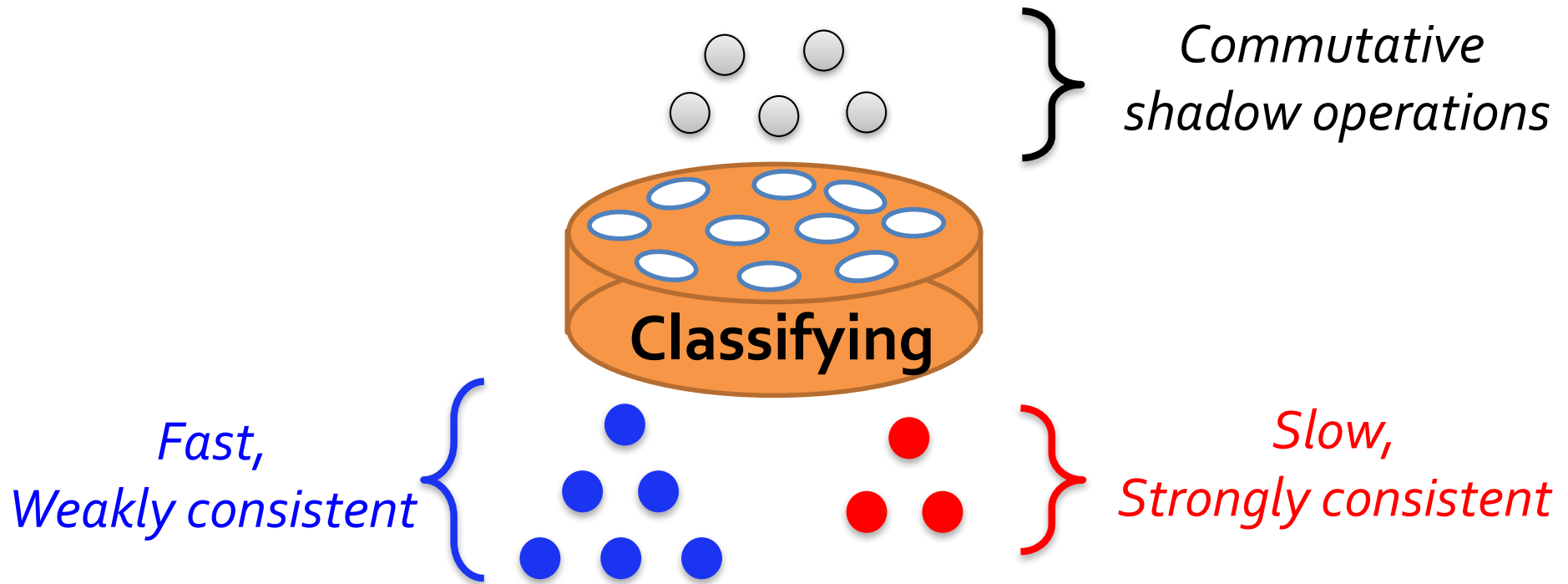
```
@AUSER CREATE TABLE BankAccount(  
    id INT(11) NOT NULL,  
    @NUMDELTA balance INT(11) default 0,  
    @LWW      name char(60) default NULL,  
    PRIMARY KEY (id)  
    ) ENGINE=InnoDB
```



# SIEVE



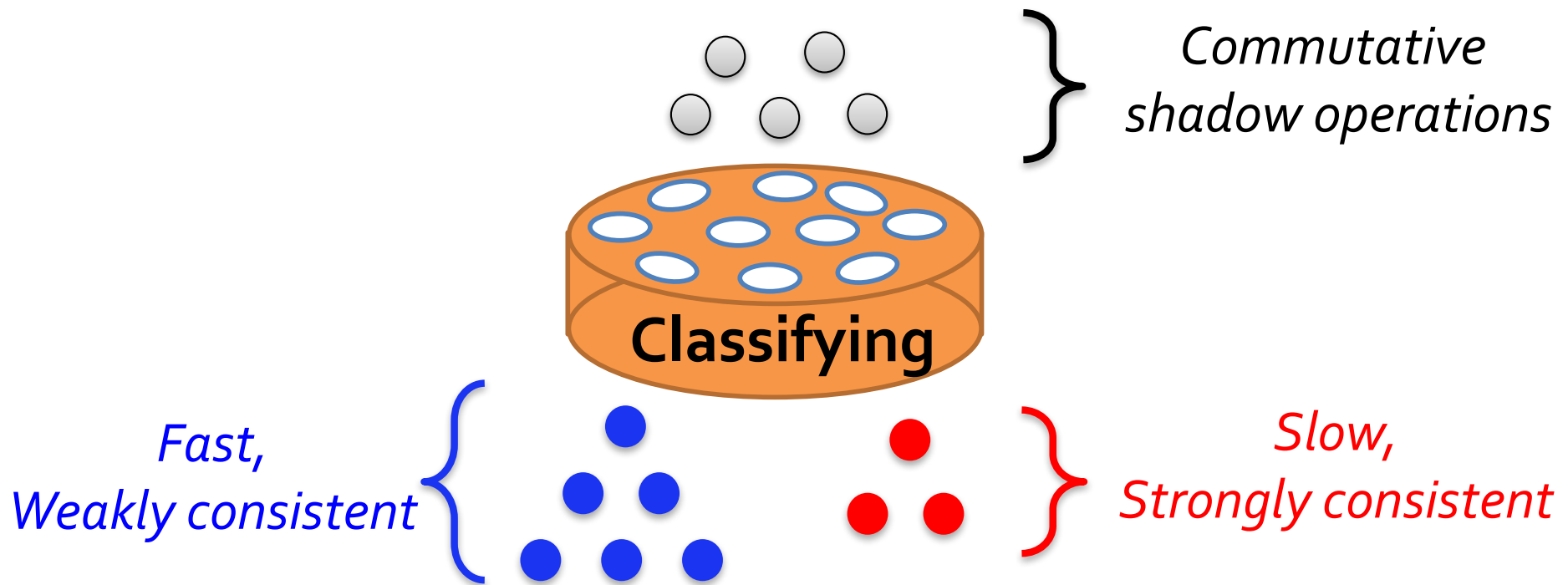
# SIEVE



# SIEVE

Challenge:

- How to classify accurately and efficiently?

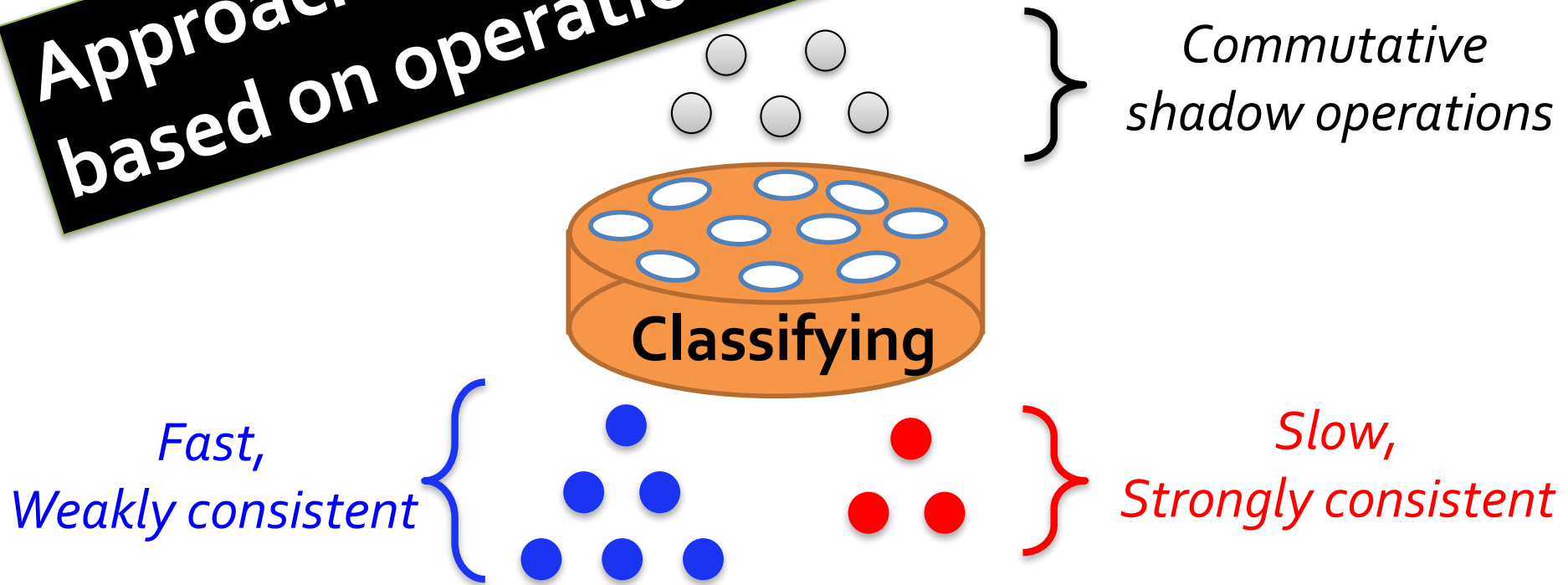


# SIEVE

Challenge:

- How to classify efficiently?

**Approach: path analysis and classification based on operation parameters**



# OUTLINE

- Context / problem
- First take: Sieve
- **Second take: SQL IPA**
- Final remarks

# Limitations of Sieve

- Operations that may violate the invariant need to be red/coordinated  $\Rightarrow$  slow
  - Acquiring reservation/token (Indigo/CISE)
- Static analysis of complete application(s)
  - Changes in applications require rerunning the analysis process

# Limitations of Sieve

- Operations that may violate the invariant need to be blue/coordinated  $\Rightarrow$  slow
  - Acquiring reservation/token (Indigo/CISE)

**Goal:** maintain invariants while avoiding coordination

# GEO-REPLICATION



removeTournament(A):  
 delete from tournament where id = 'A'  
 delete from PT where t = 'A'

Player	PT	Tournament
Sonic	Sonic, A	A
Pacman	Sonic, B	B
Mario		

Player	PT	Tournament
Sonic	<del>Sonic, A</del>	<del>A</del>
Pacman	Sonic, B	B
Mario		



# GEO-REPLICATION



enroll(Mario, A):  
insert into PT values('Mario','A')


Player	PT	Tournament
Sonic	Sonic, A	A
Pacman	Sonic, B	B
Mario	Mario, A	

Player	PT	Tournament
Sonic	<del>Sonic, A</del>	<del>A</del>
Pacman	Sonic, B	B
Mario		

# GEO-REPLICATION



```
enroll(Mario, A):
insert into PT values('Mario','A')
touch tournament where id = 'A'
```



Player	PT	Tournament
Sonic	Sonic, A	A 
Pacman	Sonic, B	B
Mario	Mario, A	

Player	PT	Tournament
Sonic	<del>Sonic, A</del>	<del>A</del>
Pacman	Sonic, B	B
Mario		

# GEO-REPLICATION



```
enroll(Mario, A):
insert into PT values('Mario','A')
touch tournament where id = 'A'
touch cascade PT where t='A'
```

Player	PT	Tournament
Sonic	Sonic, A 	A 
Pacman	Sonic, B	B
Mario	Mario, A	



Player	PT	Tournament
Sonic	<del>Sonic, A</del>	<del>A</del>
Pacman	Sonic, B	B
Mario		



# GEO-REPLICATION



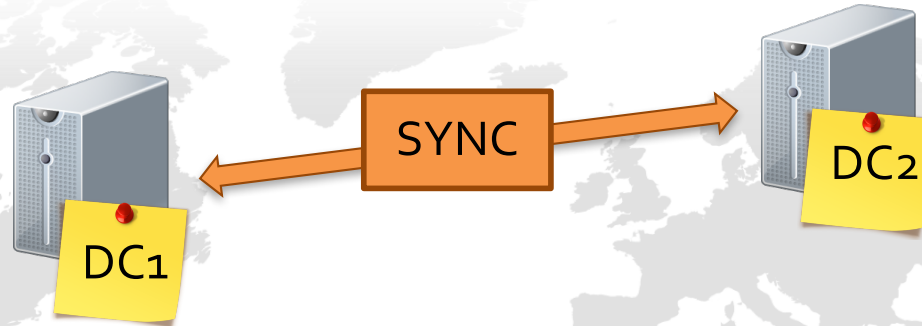
enroll(Mario, A):  
 insert into PT values('Mario','A')  
 touch tournament where id = 'A'  
 touch **cascade** PT where t='A'

removeTournament(A):  
 delete from tournament where id = 'A'  
 delete **cascade** from PT where t = 'A'







Player	PT	Tournament
Sonic	Sonic, A 	A 
Pacman	Sonic, B	B
Mario	Mario, A	



Player	PT	Tournament
Sonic	Sonic 	A 
Pacman	Sonic, B	B
Mario		



# GEO-REPLICATION



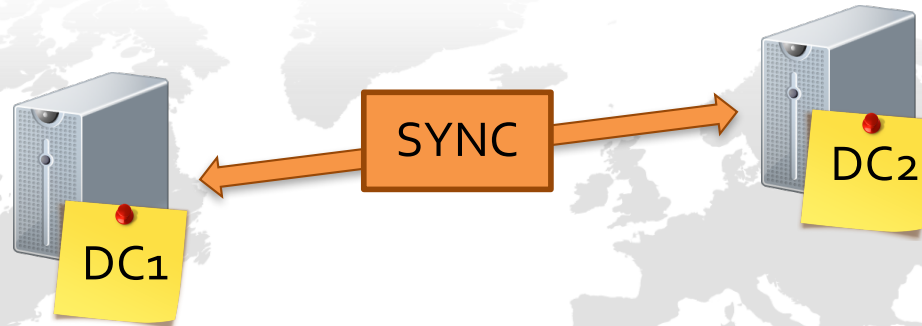
## Rules add-wins:

1.  ||  => 
2.  ||  => 







Player	PT	Tournament
Sonic	Sonic, A 	A 
Pacman	Sonic, B	B
Mario	Mario, A	

Player	PT	Tournament
Sonic	Sonic 	A 
Pacman	Sonic, B	B
Mario		

# GEO-REPLICATION



## Rules add-wins:

1.  ||  => 
2.  ||  => 

Player	PT	Tournament
Sonic	Sonic, A	A
Pacman	Sonic, B	B
Mario	Mario, A	

Player	PT	Tournament
Sonic	Sonic, A	A
Pacman	Sonic, B	B
Mario		

# Other invariants

- Primary key (uniqueness)
  - Split keyspace
- Check constraint
  - E.g. stock int CHECK (stock  $\geq$  0)
  - Solved using bounded counter (escrow)

# Limitations of Sieve

**Goal:** “modify” operations in runtime.  
Use schema definition.

- Static analysis of complete application(s)
  - Changes in applications require rerunning the analysis process



```

create table player( id varchar(20), primary key id)
create table tournament( id varchar(20), primary key id)
create table pt( p varchar(20), t varchar(20), foreign key (p) REFERENCES
player (id), foreign key (t) REFERENCES tournament (id))
  
```



```

enroll(Mario, A):
insert into PT values('Mario','A')
  
```

Player	PT	Tournament
Sonic	Sonic, A	A
Pacman	Sonic, B	B
Mario	Mario, A	

Player	PT	Tournament
Sonic	<del>Sonic, A</del>	<del>A</del>
Pacman	Sonic, B	B
Mario		

```

create table player( id varchar(20), primary key id)
create table tournament( id varchar(20), primary key id)
create table pt( p varchar(20), t varchar(20), AW foreign key (p)
REFERENCES player (id), AW foreign key (t) REFERENCES tournament (id))
  
```




```

enroll(Mario, A):
insert into PT values('Mario','A')
touch tournament where id = 'A'
touch player where id = 'Mario'
  
```

Player
Sonic
Pacman
Mario 

PT
Sonic, A
Sonic, B
Mario, A

Tournament
A 
B

Player
Sonic
Pacman
Mario

PT
<del>Sonic, A</del>
Sonic, B

Tournament
<del>A</del>
B

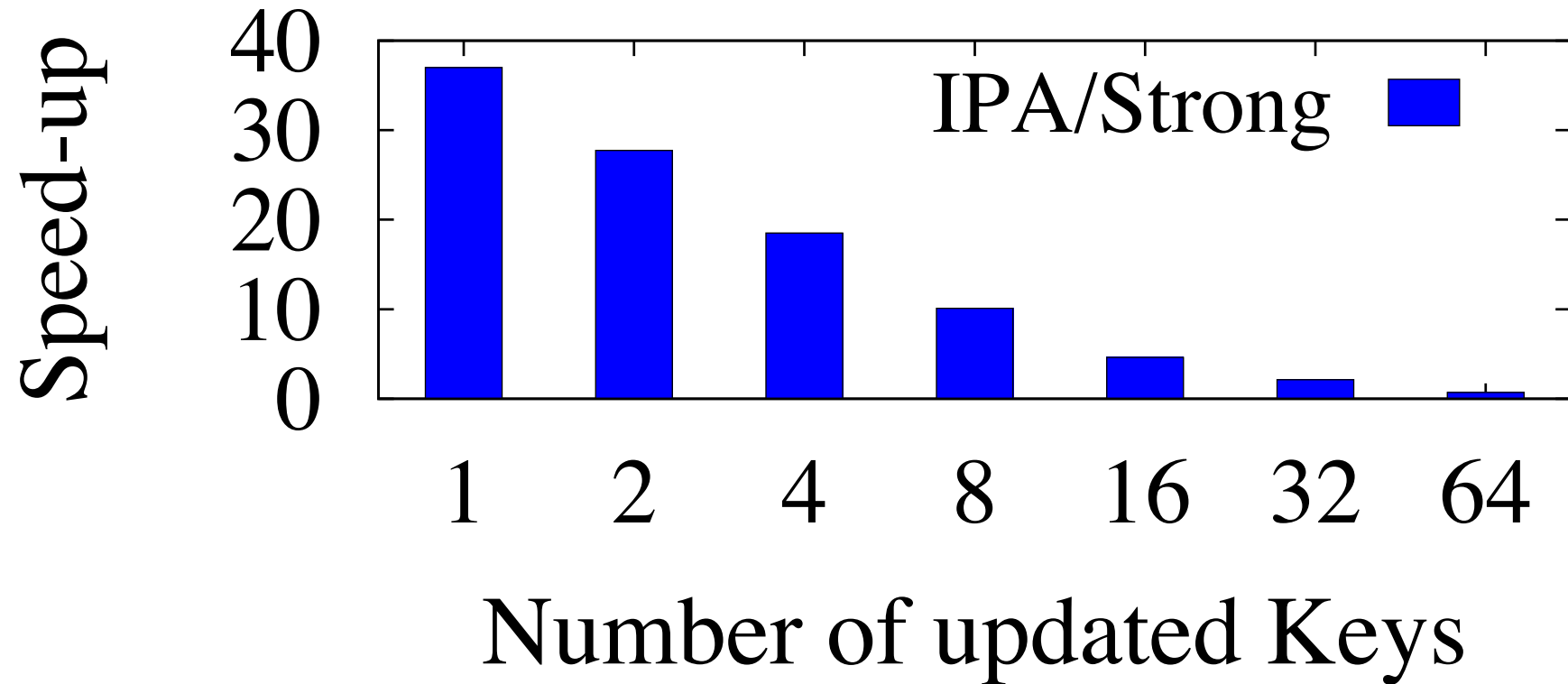
# OUTLINE

- Context / problem
- First take: Sieve
- Second take: SQL IPA
- **Final remarks**

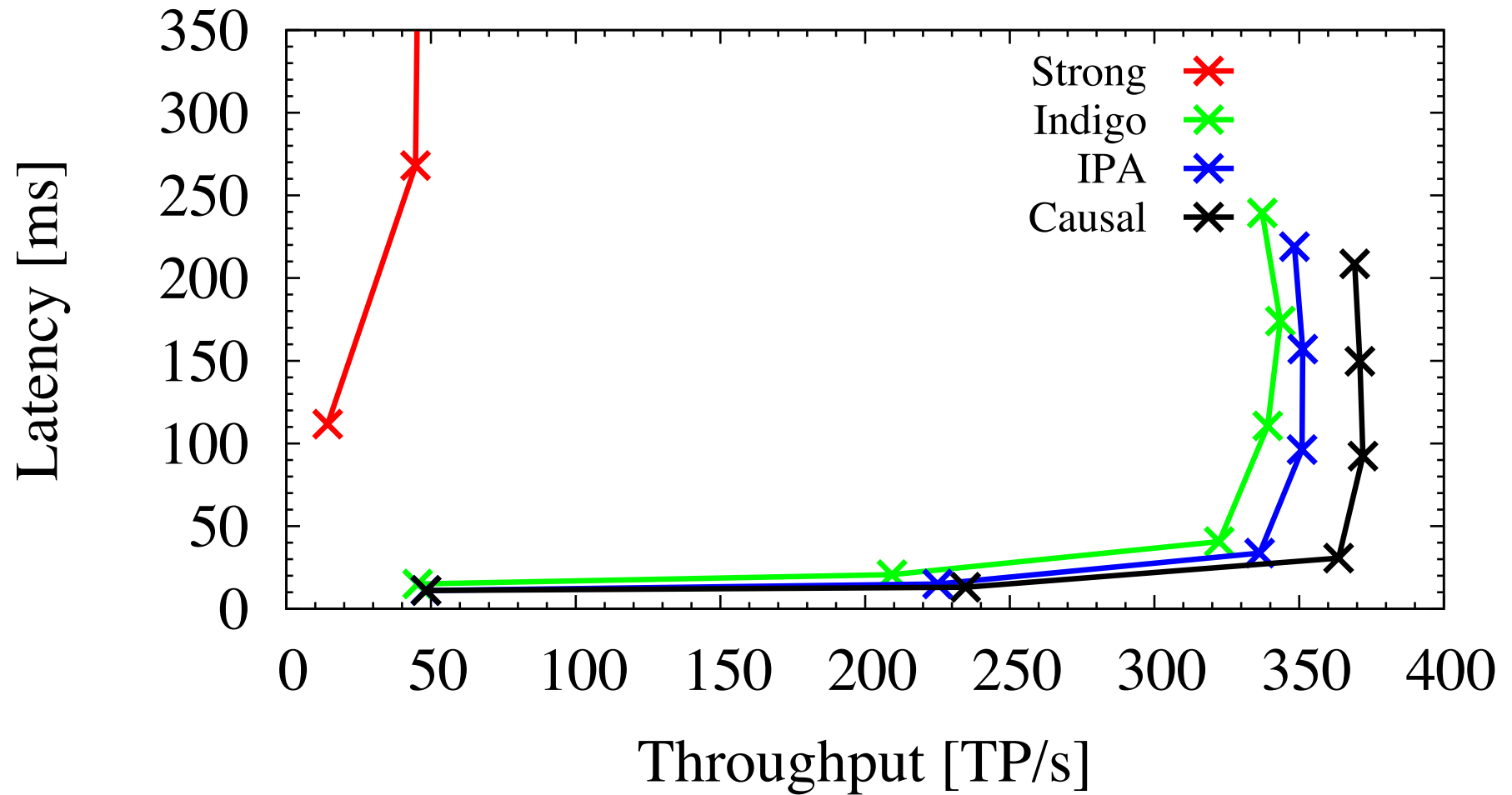
# Status

- Implementing prototype on top of Antidote database
- Runtime solution equivalent to static solution implemented in IPA

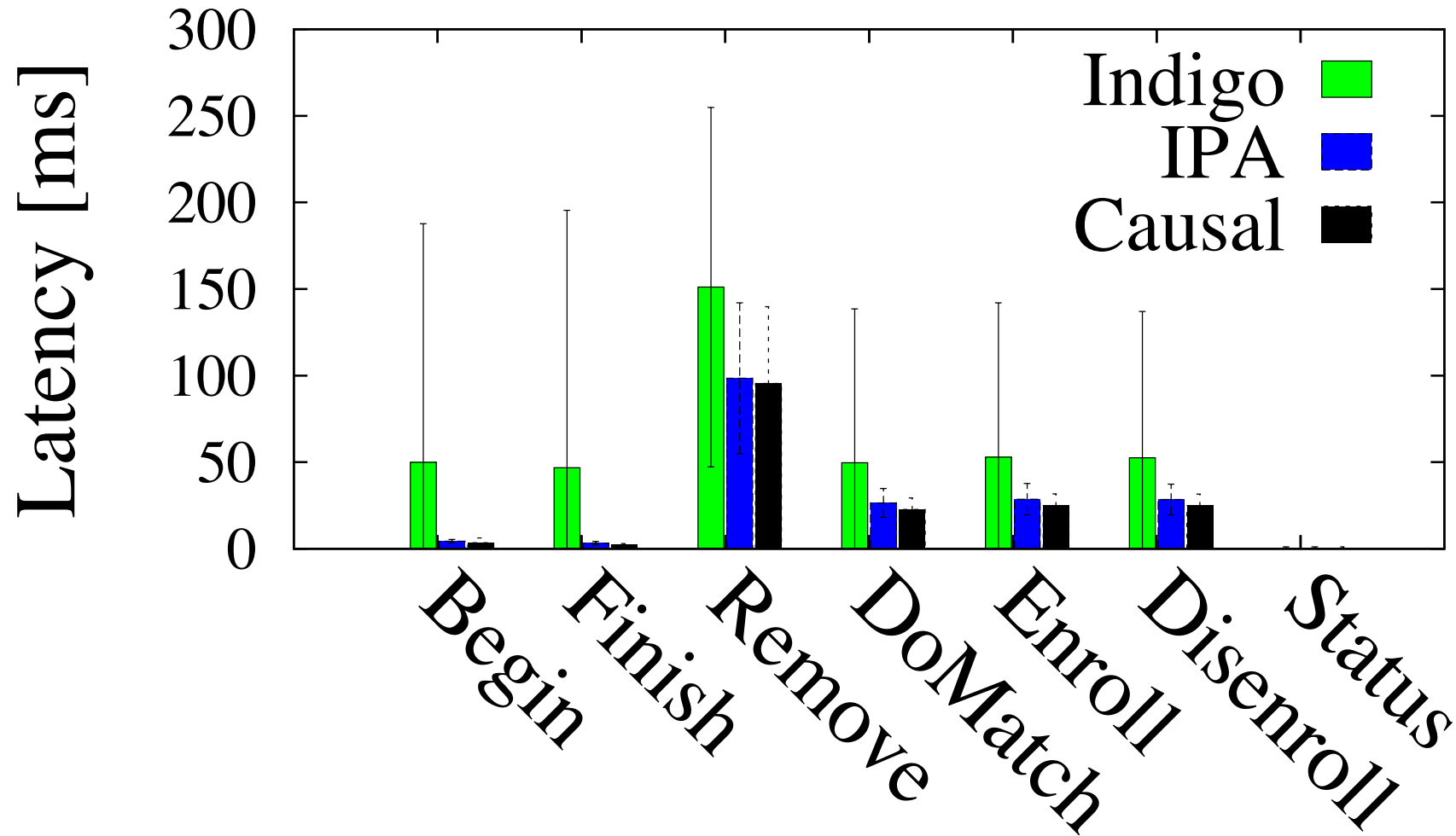
# Impact of additional updates



# TOURNAMENT



# TOURNAMENT: OPERATIONS LATENCY



# Final remarks

- SQL schema allows to define constraints
- First approach
  - Coordinate on operations that may break invariants
- Second approach
  - Maintain invariants without coordination (or minimizing coordination)





# QUESTIONS?