Salt

Combining ACID and BASE in a distributed database

Chao Xie, Chunzhi Su, Manos Kapritsos, Yang Wang, Navid Yaghmazadeh, Lorenzo Alvisi, Prince Mahajan

The University of Texas at Austin

TRANSACTIONS ARE GREAT

Four properties in a single abstraction

Atomicity Consistency Isolation Durability (**ACID**)

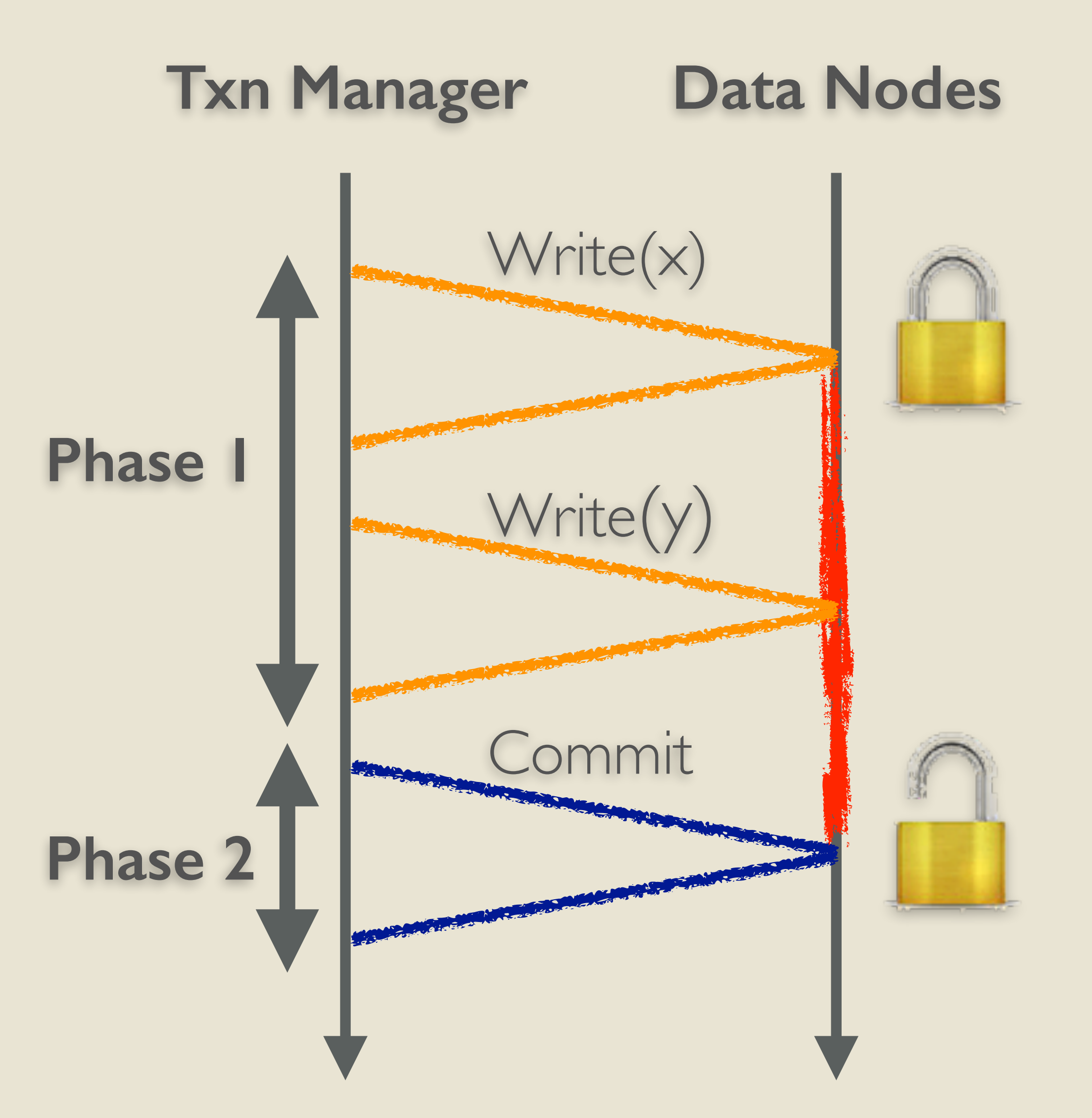
- Ease of programming
- Easy to reason about

Transaction

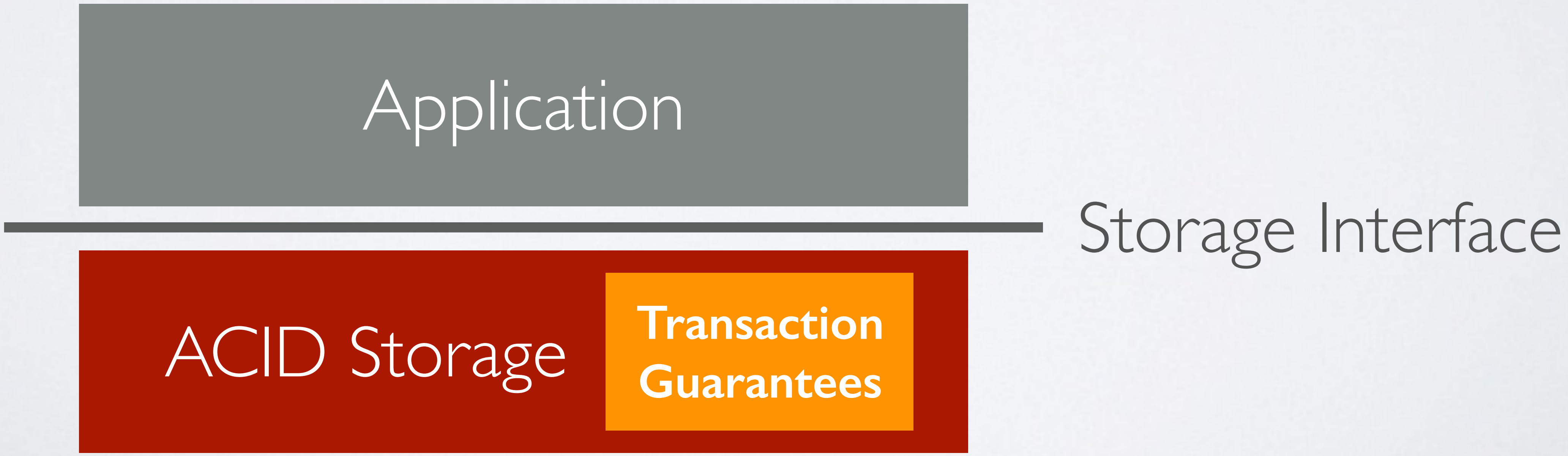
TRANSACTIONS ARE ~~GREAT~~ slow

Concurrency control limits performance

2PC protocol is costly



THE ALTERNATIVE: BASE



Application

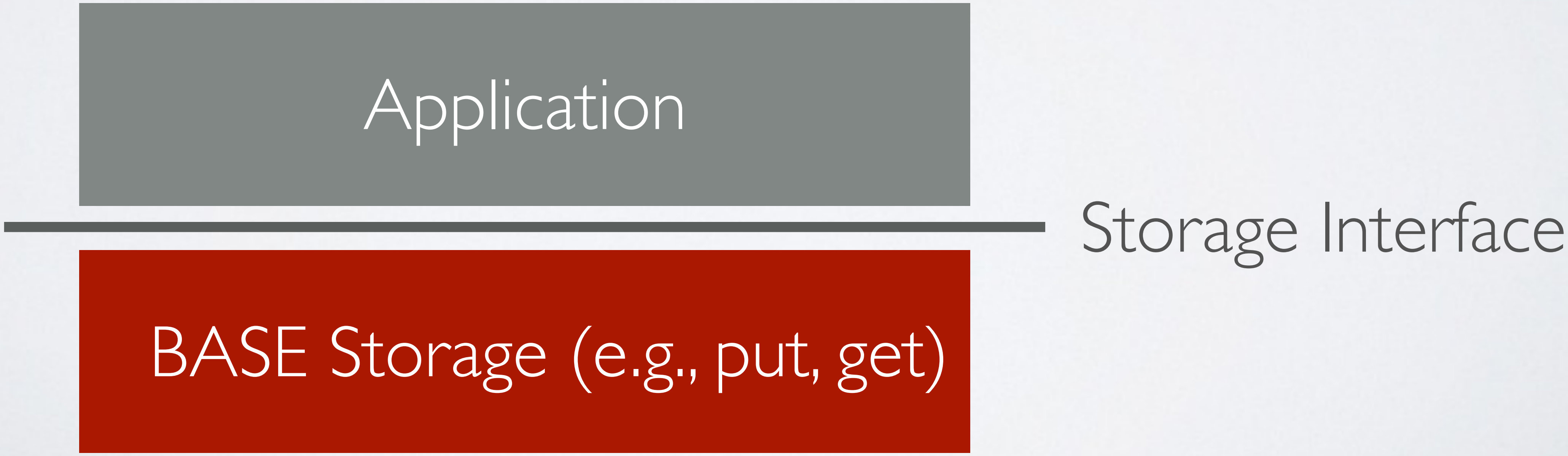
Storage Interface

ACID Storage

Transaction Guarantees

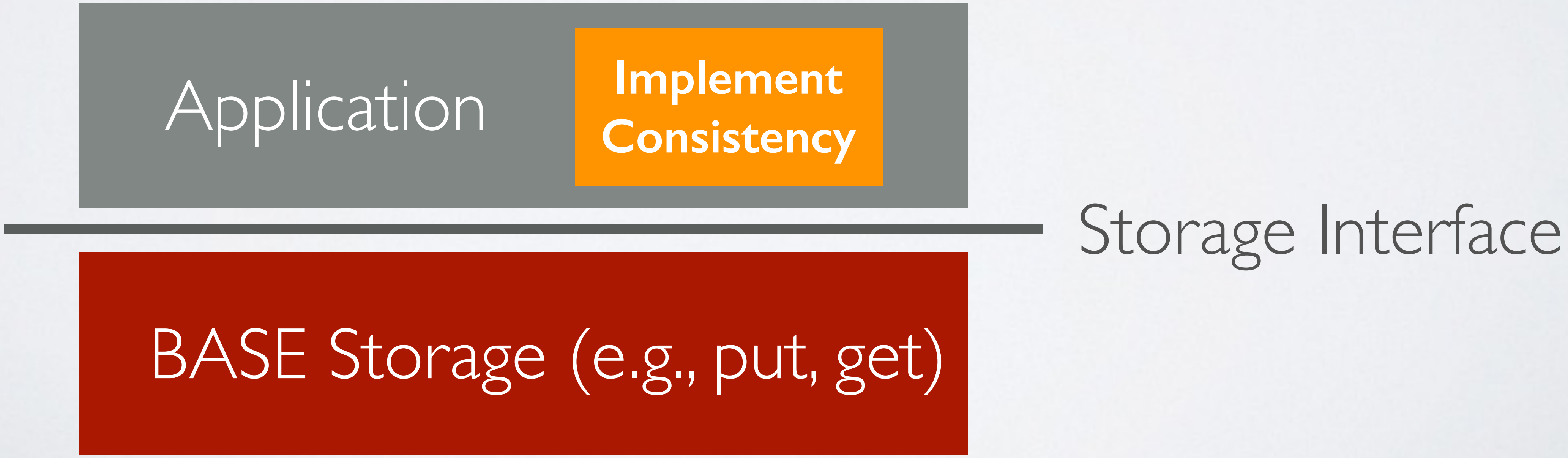
THE ALTERNATIVE: BASE

- Write custom code to get better performance

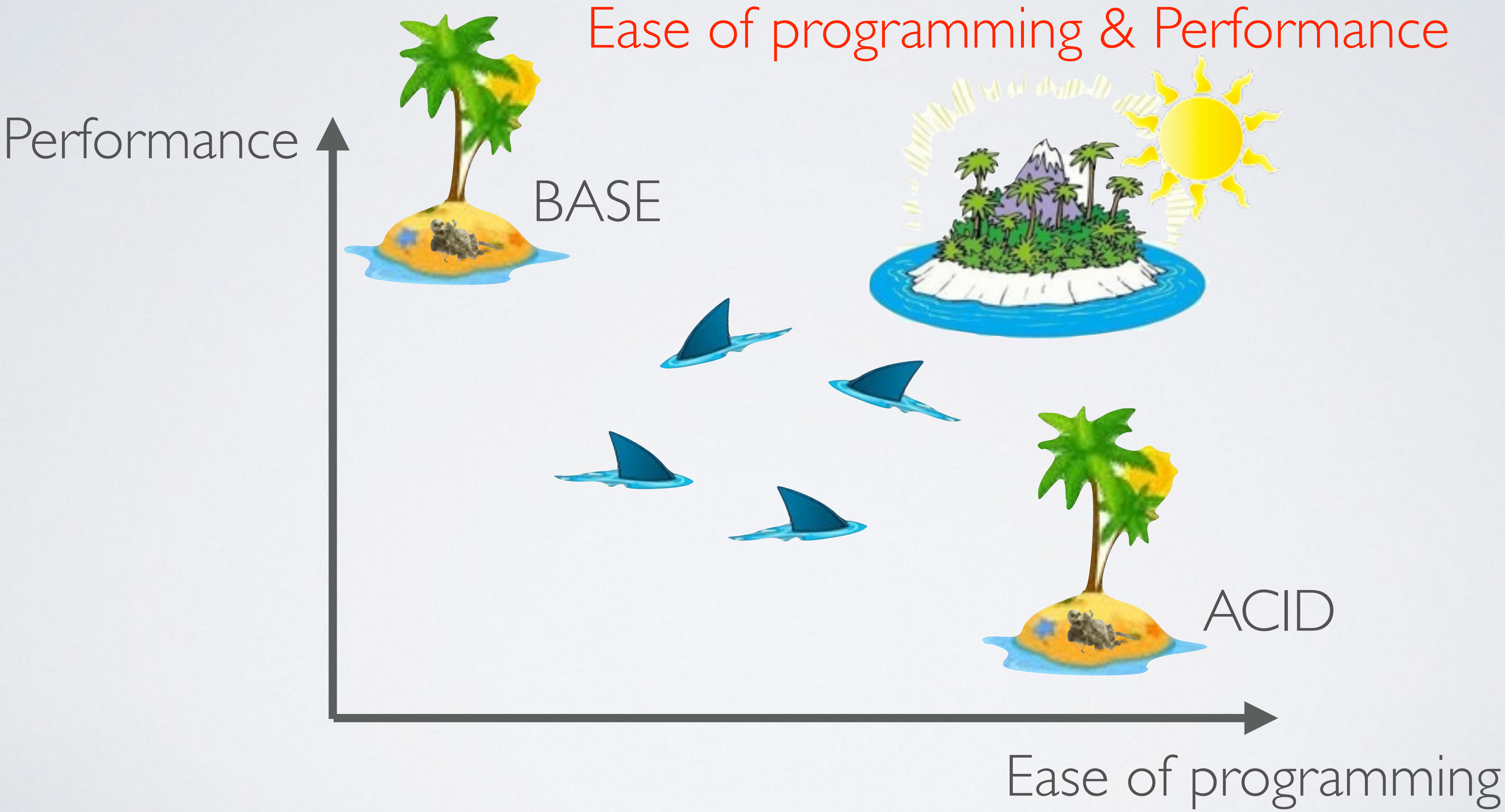


THE ALTERNATIVE: BASE

- Write custom code to get better performance
- Complexity gets out of control



A STARK CHOICE



Vilfredo Pareto

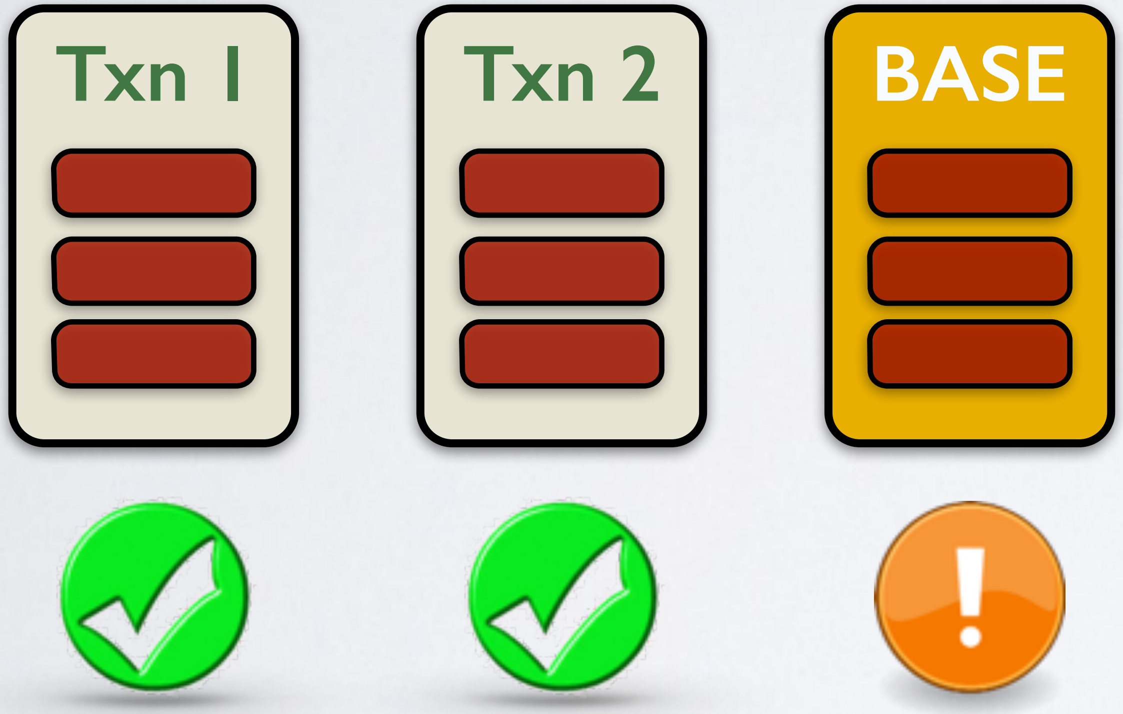
20% of the causes account for 80% of the effects

NOT ALL TRANSACTIONS ARE CREATED EQUAL

20% of the causes
account for
80% of the effects

- Many transactions are not run frequently
- Many transactions are lightweight

AN OPPORTUNITY



- Identify critical transactions
- BASE-ify only critical transactions

SALT

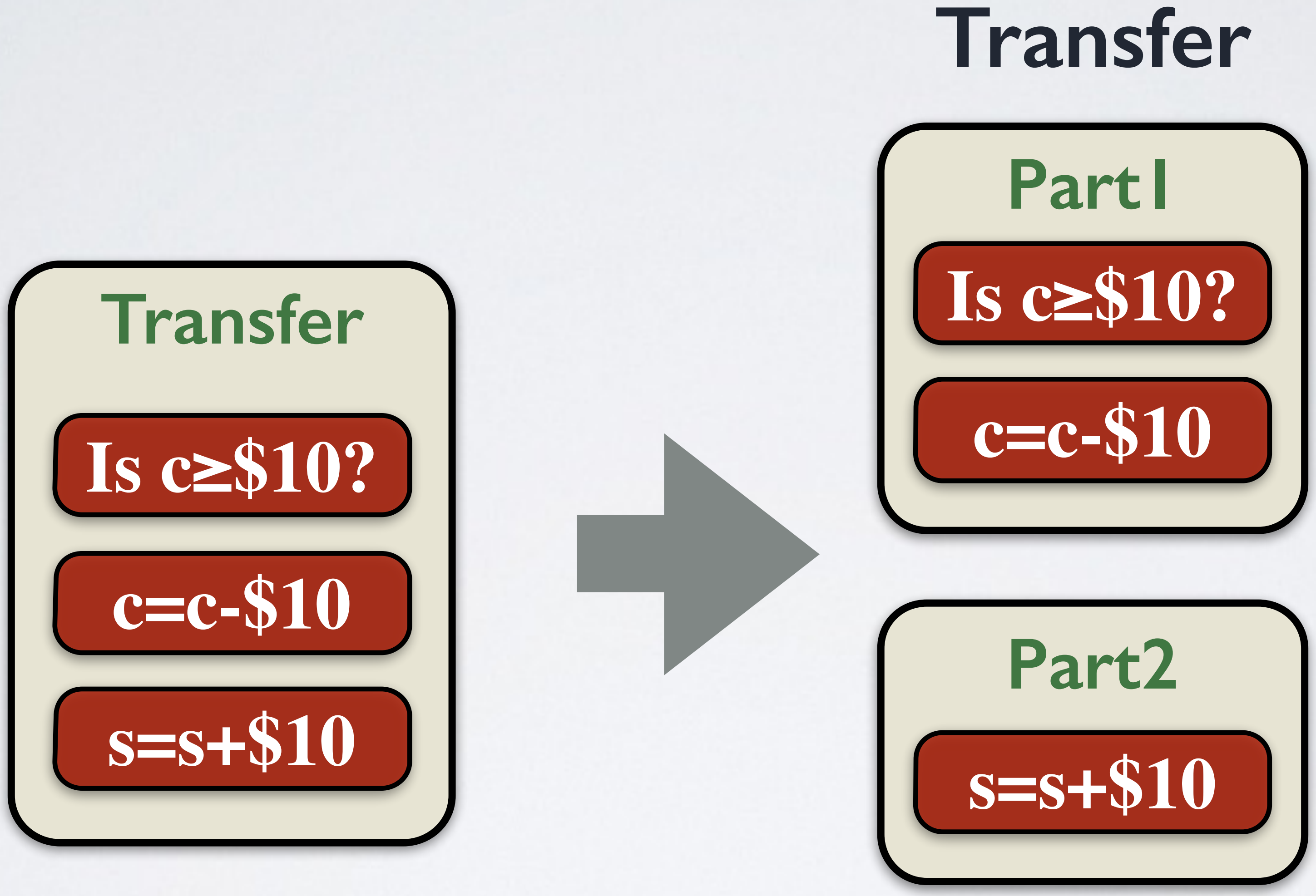
Motivation

Base Transactions & Salt Isolation

Achieving Salt Isolation

Evaluation

MORE CONCURRENCY !



Transfer

Is c≥$10?

c=c-$10

s=s+$10

Transfer

Part1

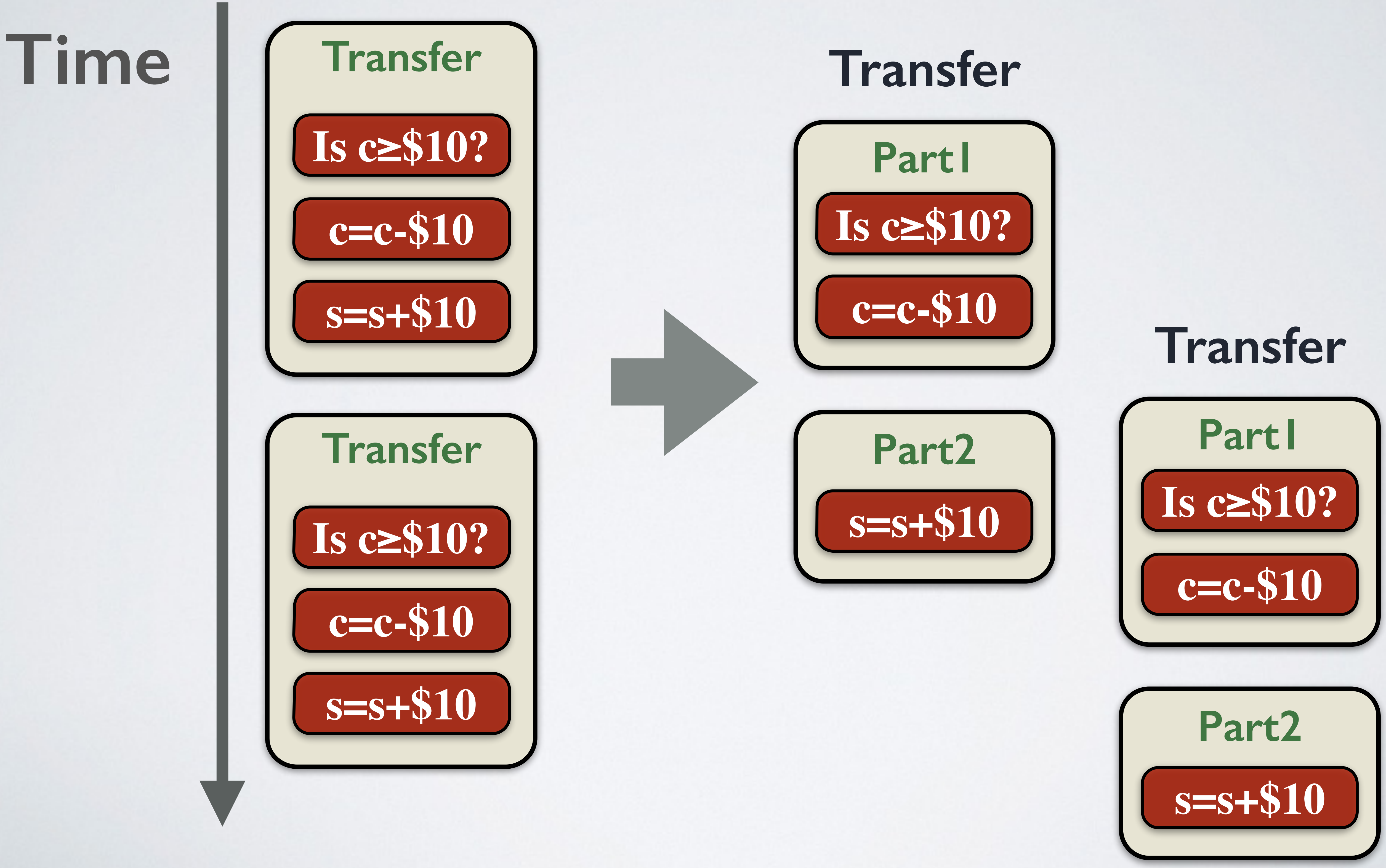
Is c≥$10?

c=c-$10

Part2

s=s+$10

MORE CONCURRENCY !



Time

Transfer

Is c≥$10?

c=c-$10

s=s+$10

Transfer

Is c≥$10?

c=c-$10

s=s+$10

Transfer

Part1

Is c≥$10?

c=c-$10

Part2

s=s+$10

Transfer

Part1

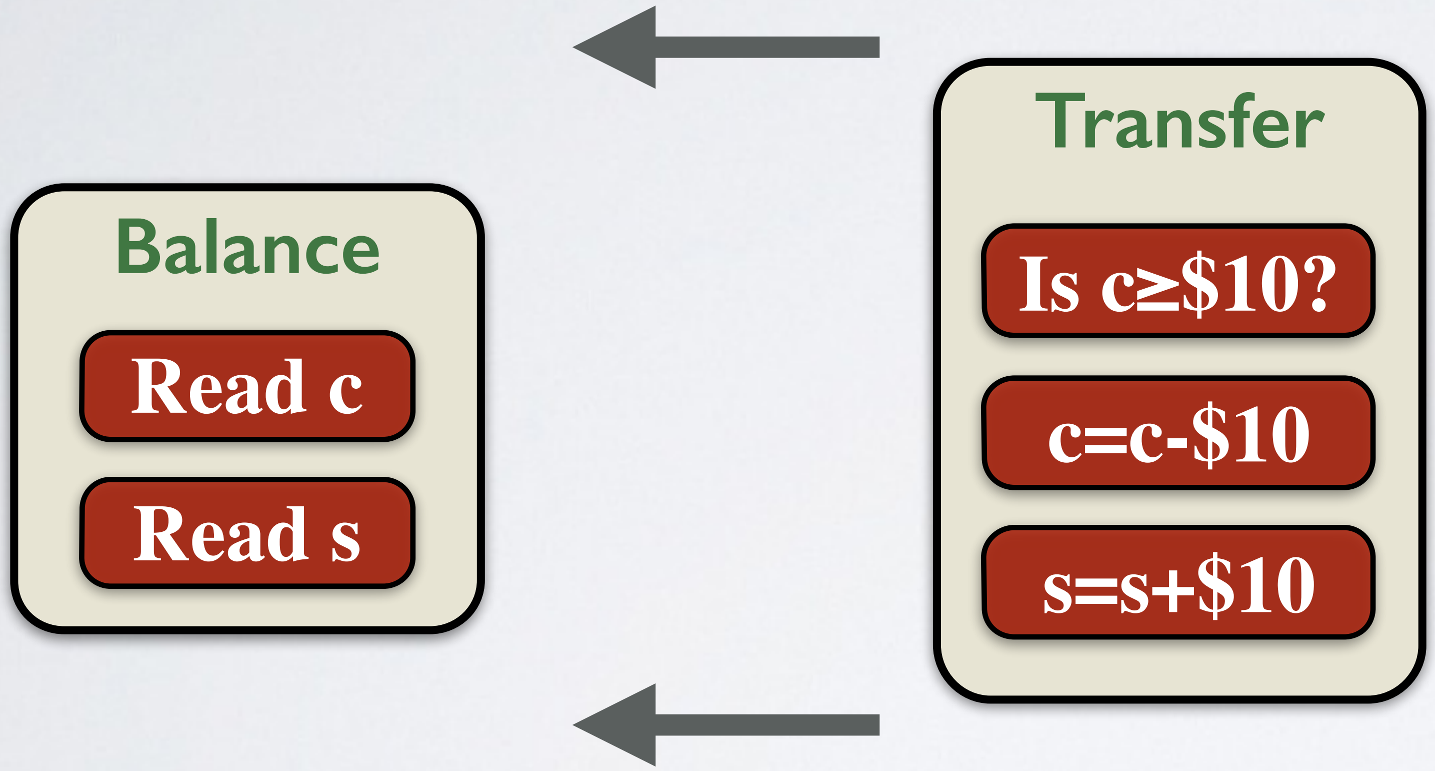
Is c≥$10?

c=c-$10

Part2

s=s+$10

CORRECTNESS AT RISK



Balance

Read c

Read s

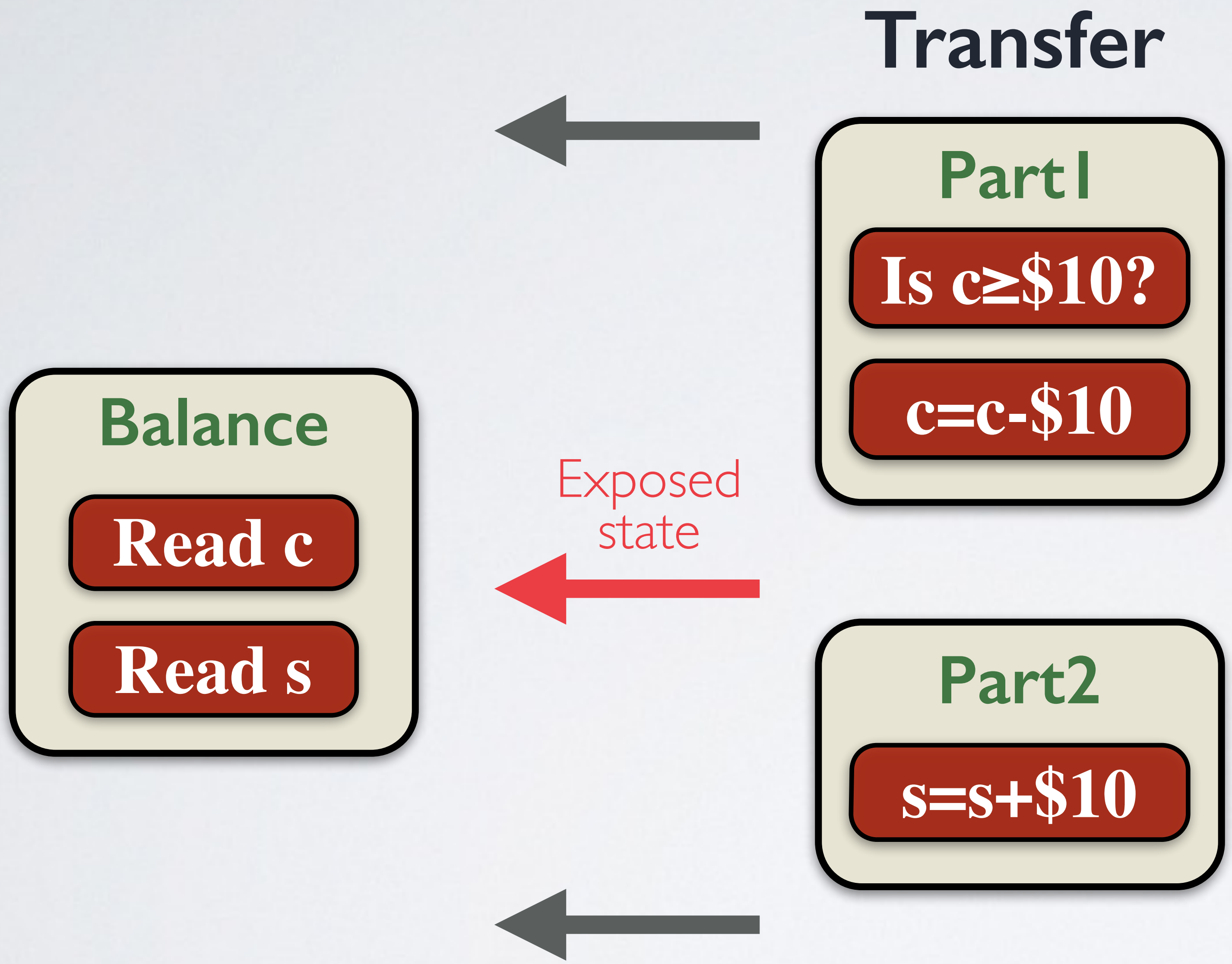
Transfer

Is c≥$10?

c=c-$10

s=s+$10

CORRECTNESS AT RISK



Transfer

Part1

Is c≥$10?

c=c-$10

Part2

s=s+$10

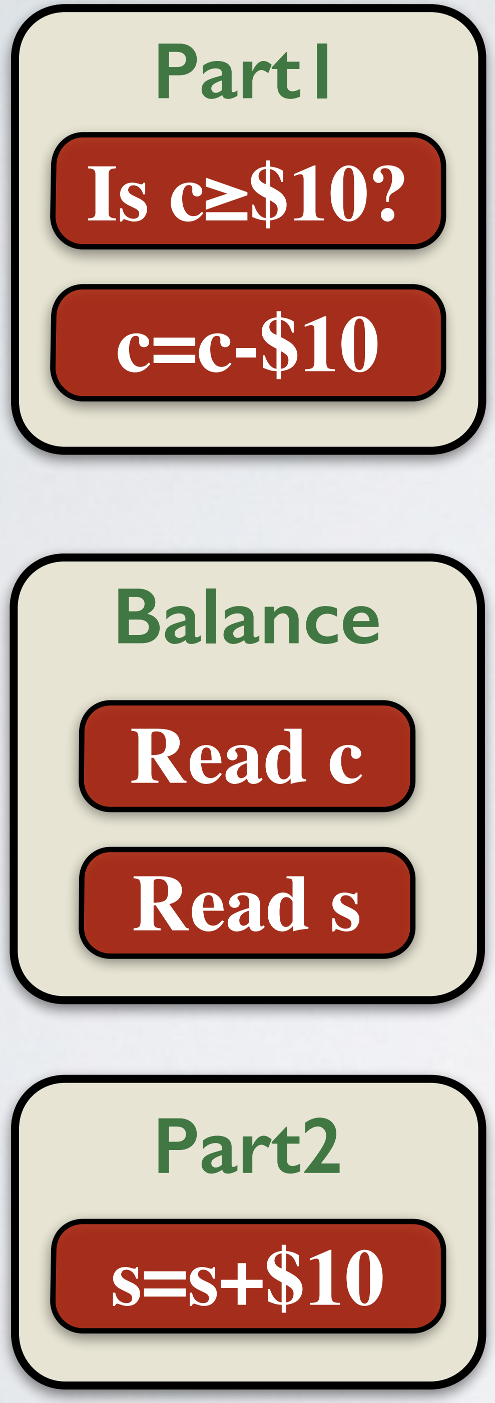
Balance

Read c

Read s

Exposed state

CORRECTNESS AT RISK



Part1

Is c≥$10?

c=c-$10

Balance

Read c

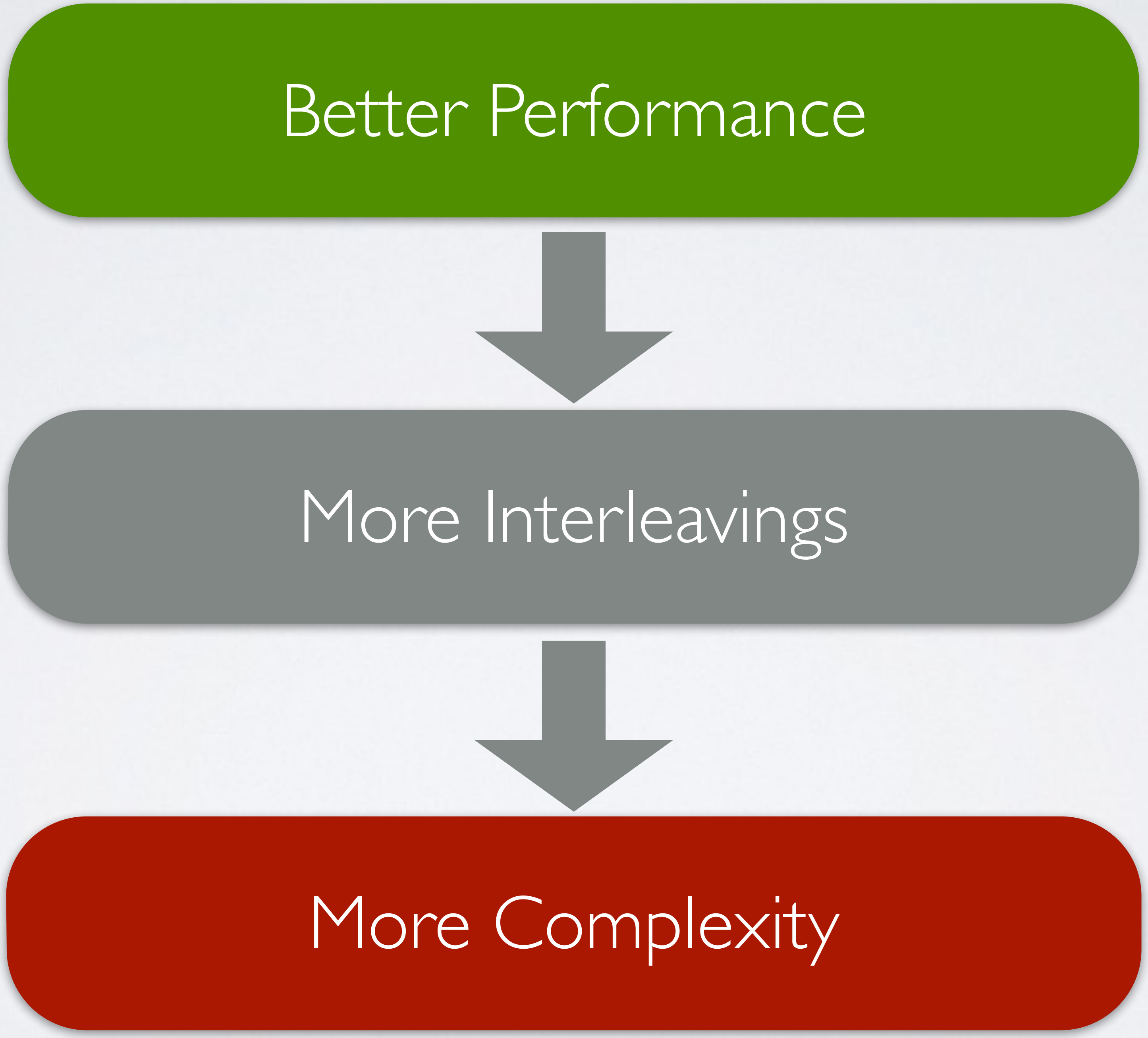
Read s

Part2

s=s+$10

Finer Isolation for one transaction may affect all transactions!!

Performance vs Complexity

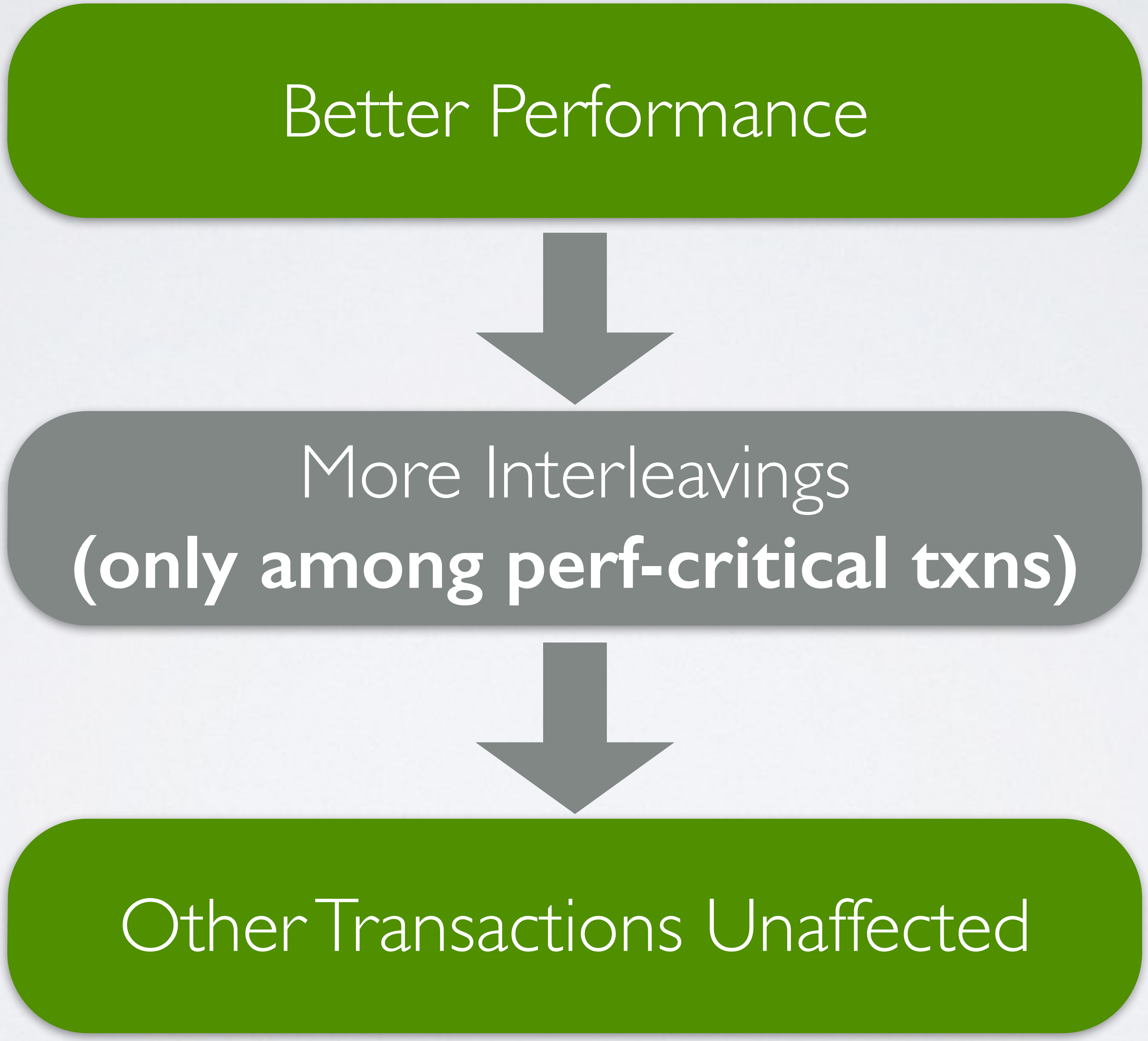


Better Performance

More Interleavings

More Complexity

Performance vs Complexity

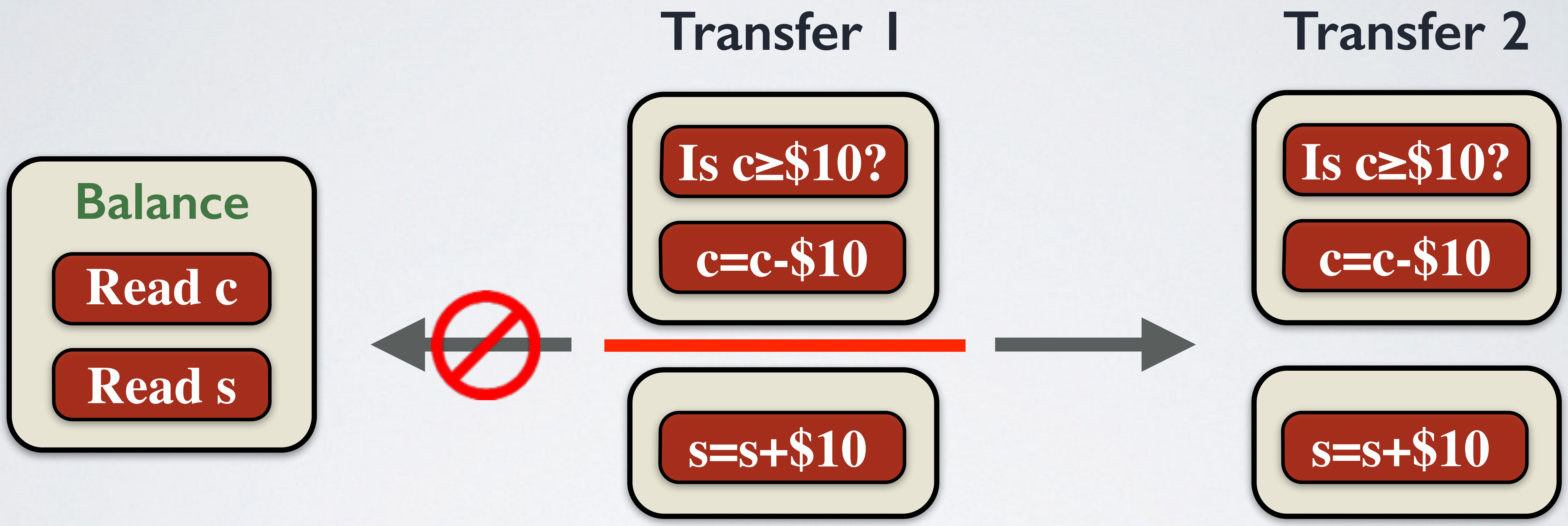


Better Performance

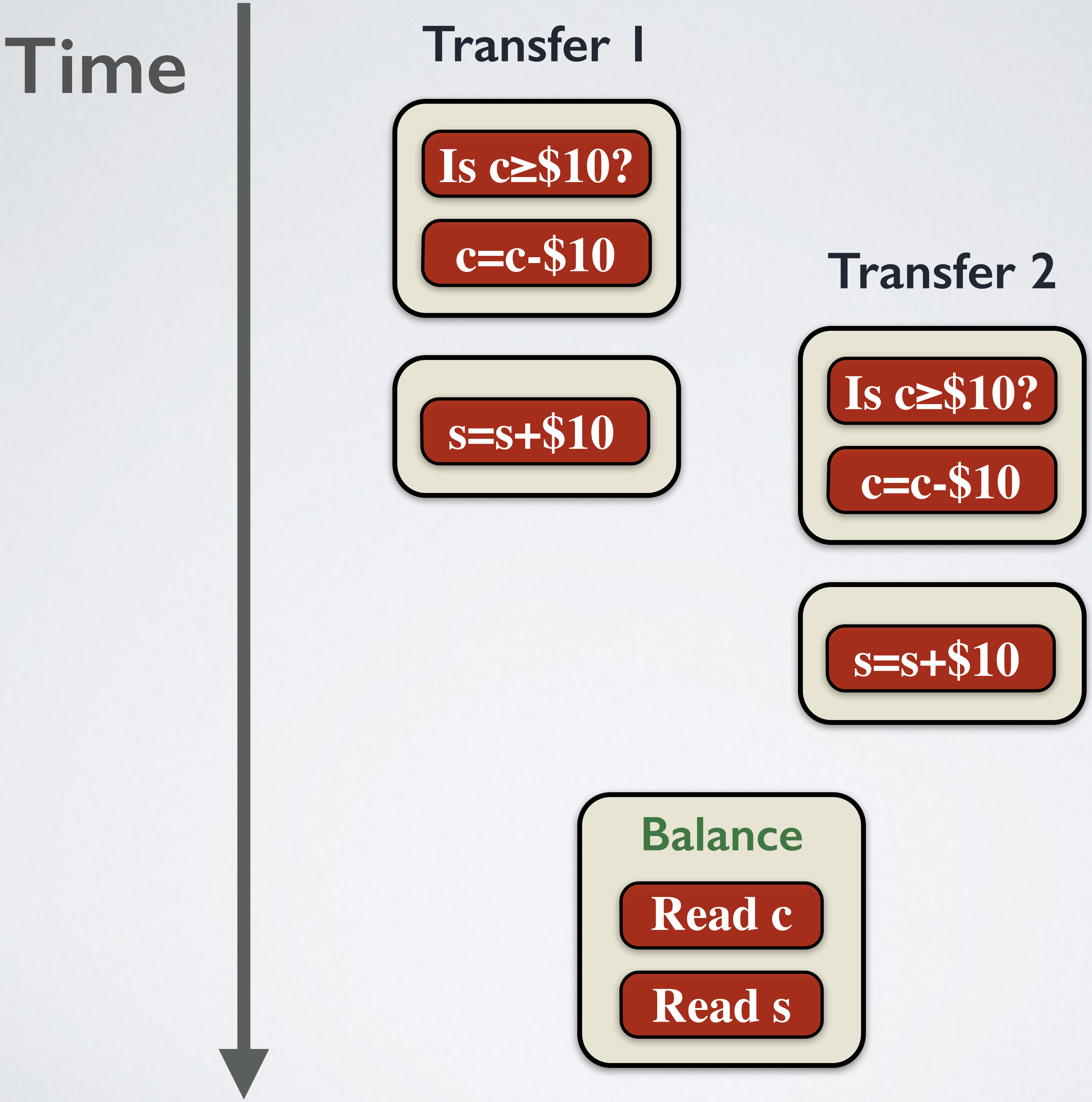
More Interleavings

(only among perf-critical txns)

Other Transactions Unaffected



Behaves differently
when interacting with different transactions



Time

Transfer 1

Is c≥$10?

c=c-$10

s=s+$10

Transfer 2

Is c≥$10?

c=c-$10

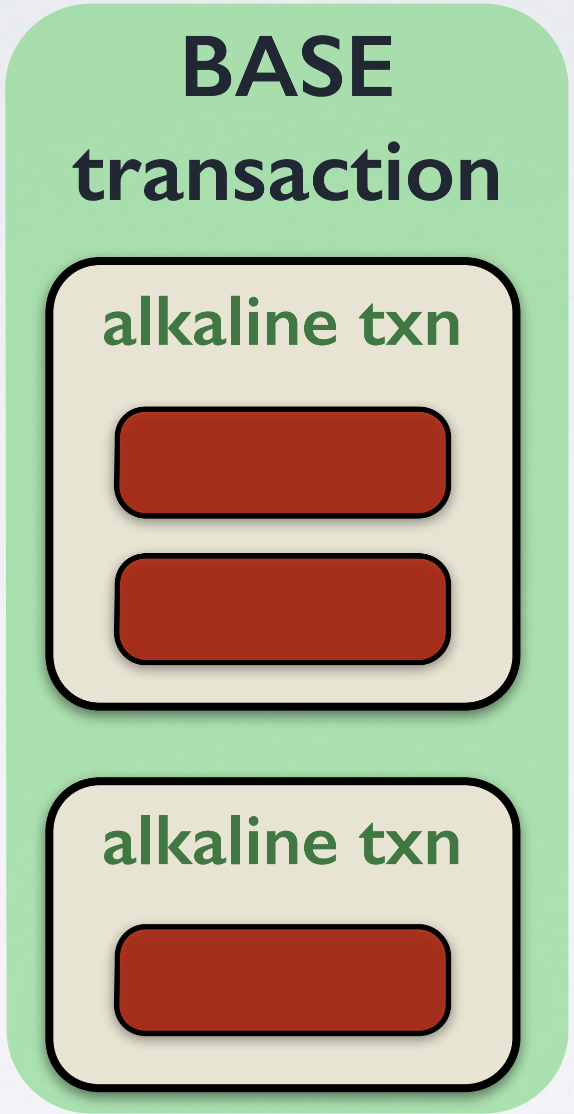
s=s+$10

Balance

Read c

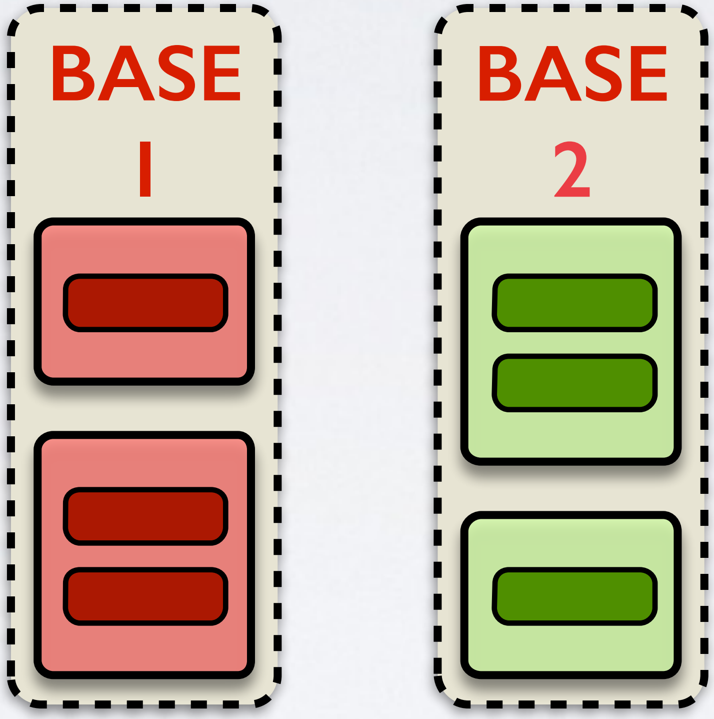
Read s

BASE TRANSACTION



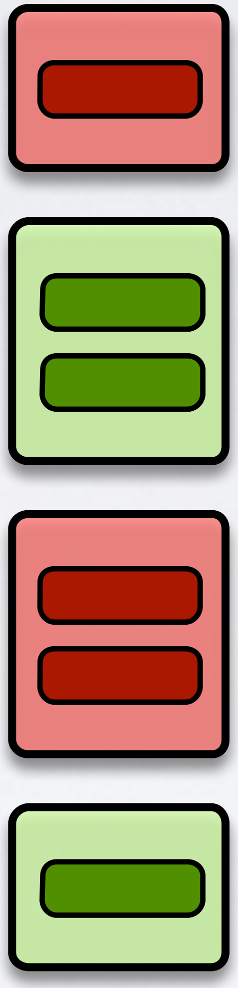
Behaves differently
when interacting with different transactions

BASE INTERACT WITH BASE



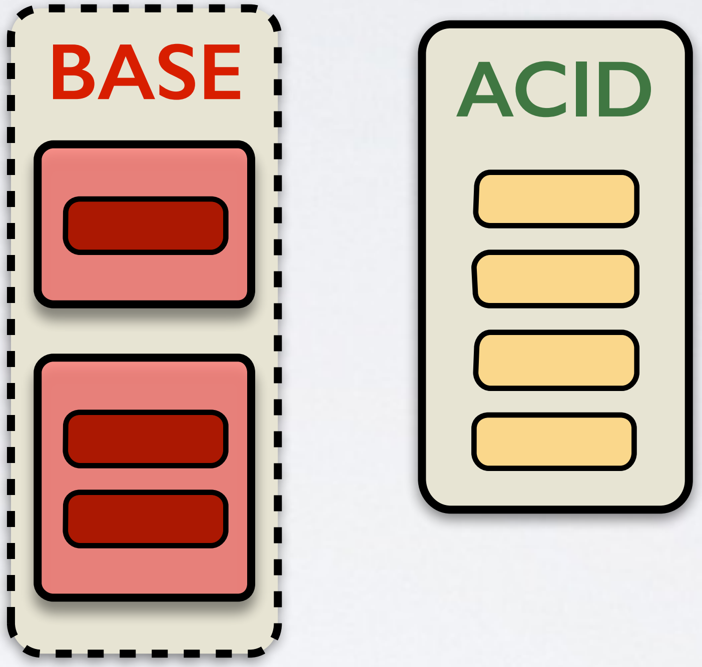
Fine Isolation granularity
between BASE transactions

BASE INTERACT WITH BASE



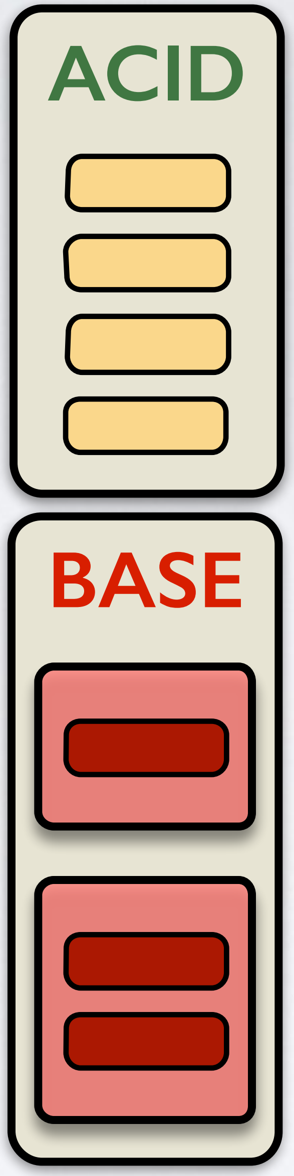
Fine Isolation granularity
between BASE transactions

BASE INTERACT WITH ACID



BASE transactions provide coarse Isolation granularity to ACID transactions

BASE INTERACT WITH ACID



BASE transactions provide coarse Isolation granularity to ACID transactions

SALT ISOLATION

BASE transactions: multiple granularities of Isolation

To BASE transactions:
a sequence of small
ACID transactions

To ACID transactions:
a single, monolithic
ACID transaction

Performance & Ease of Programming

SALT

Motivation

Base Transactions & Salt Isolation

Achieving Salt Isolation

Evaluation

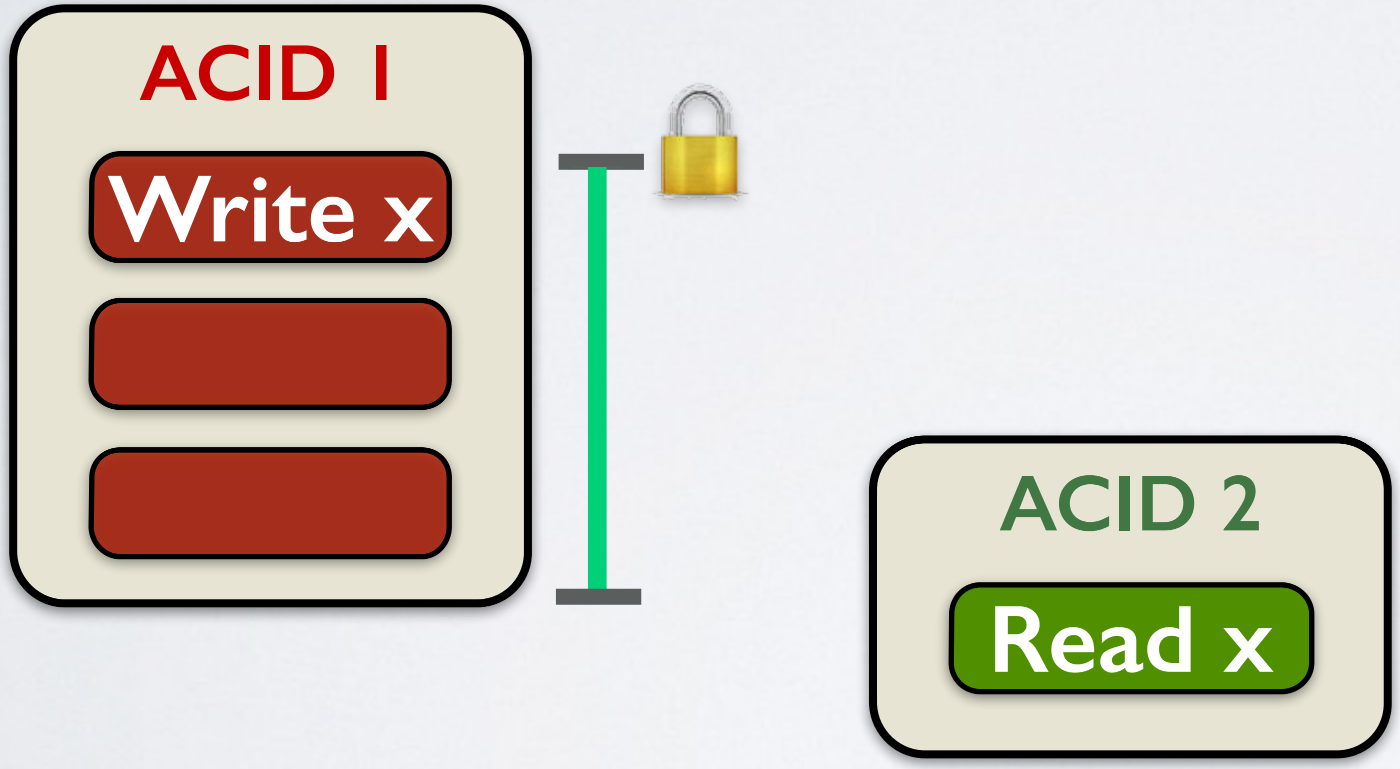
ONE MECHANISM

LOCKS

Three flavors

ACID locks
Alkaline locks
Saline locks

ACID LOCKS



ACID 1

Write x

ACID 2

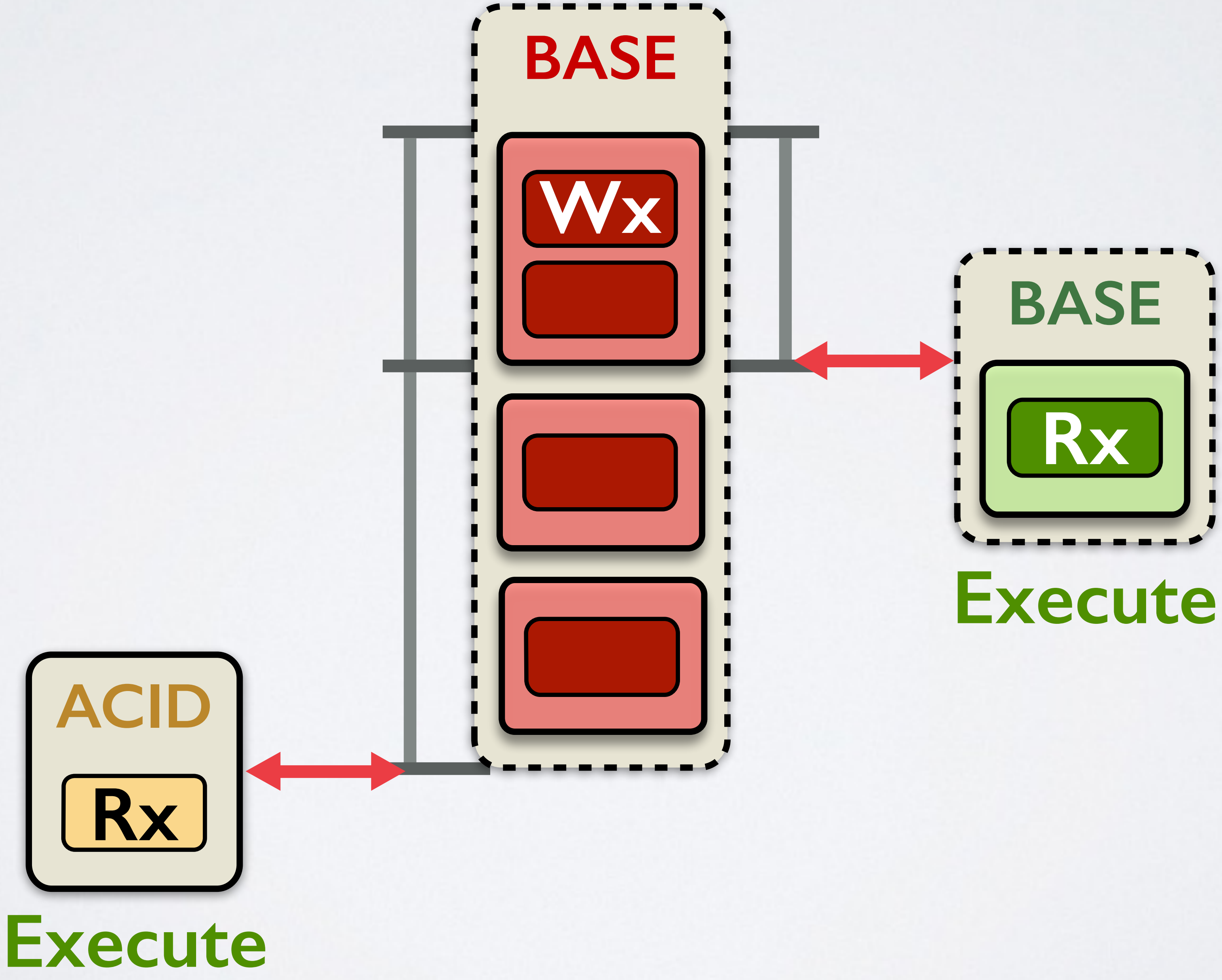
Read x

Execute

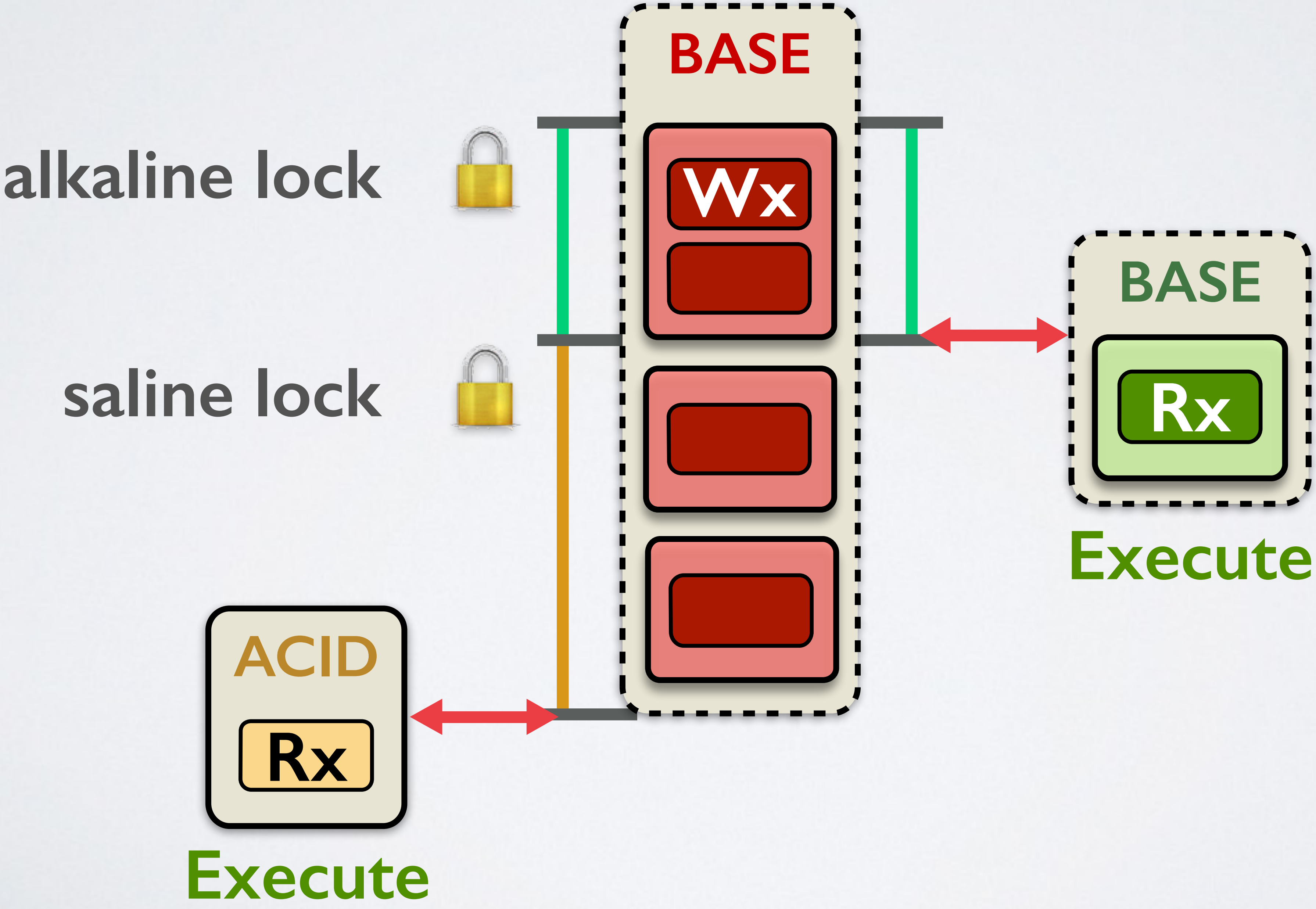
	AC-R	AC-W
AC-R	✓	✗
AC-W	✗	✗

Lock Table

LOCKS?



LOCKS?



alkaline lock

saline lock

BASE

Wx

BASE

Rx

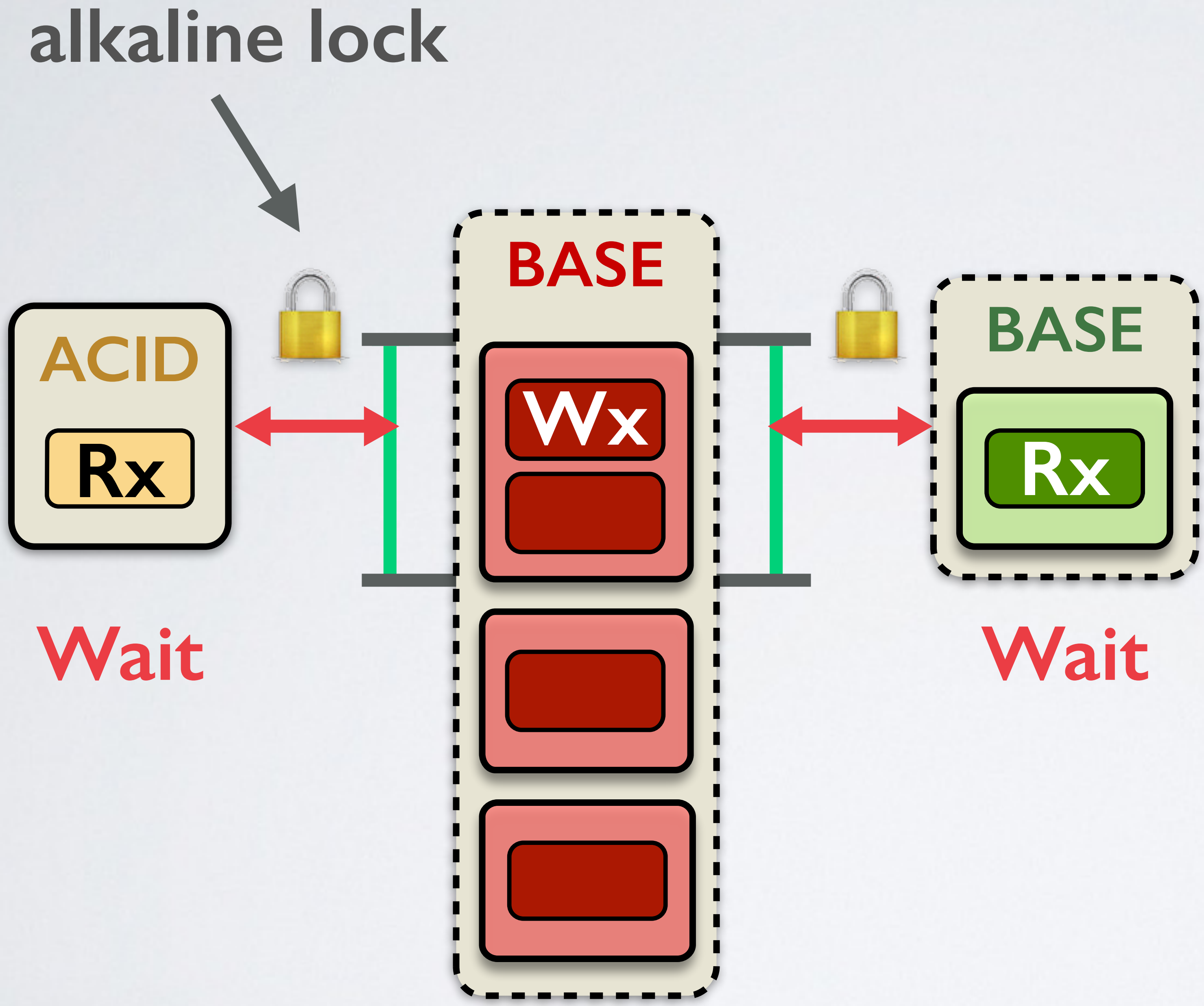
Execute

ACID

Rx

Execute

ALKALINE LOCKS



alkaline lock

ACID

Rx

Wait

BASE

Wx

BASE

Rx

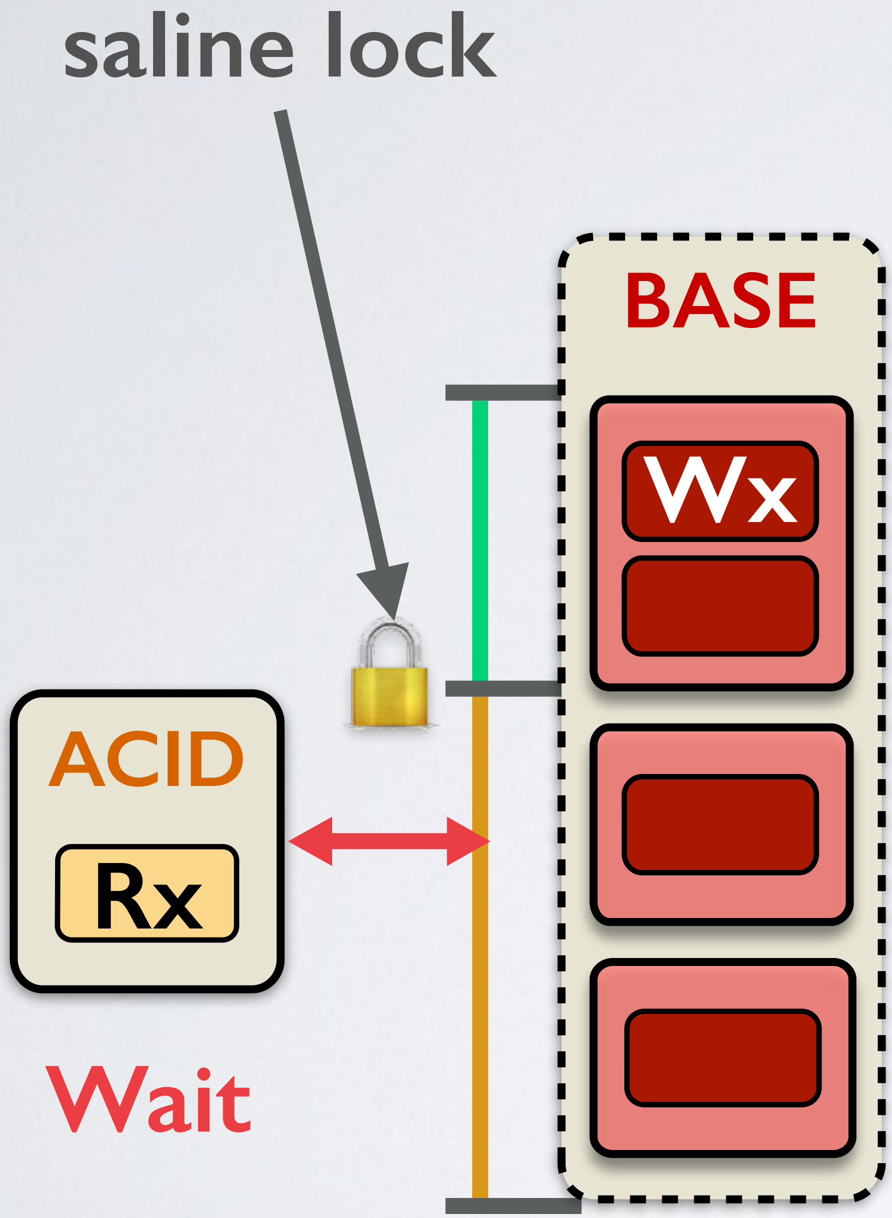
Wait

	AC-R	AC-W	alk-R	alk-W
AC-R	✓	✗	✓	✗
AC-W	✗	✗	✗	✗
alk-R	✓	✗	✓	✗
alk-W	✗	✗	✗	✗

Lock Table

Conflict with ACID & alkaline locks

SALINE LOCKS



saline lock

BASE

Wx

ACID

Rx

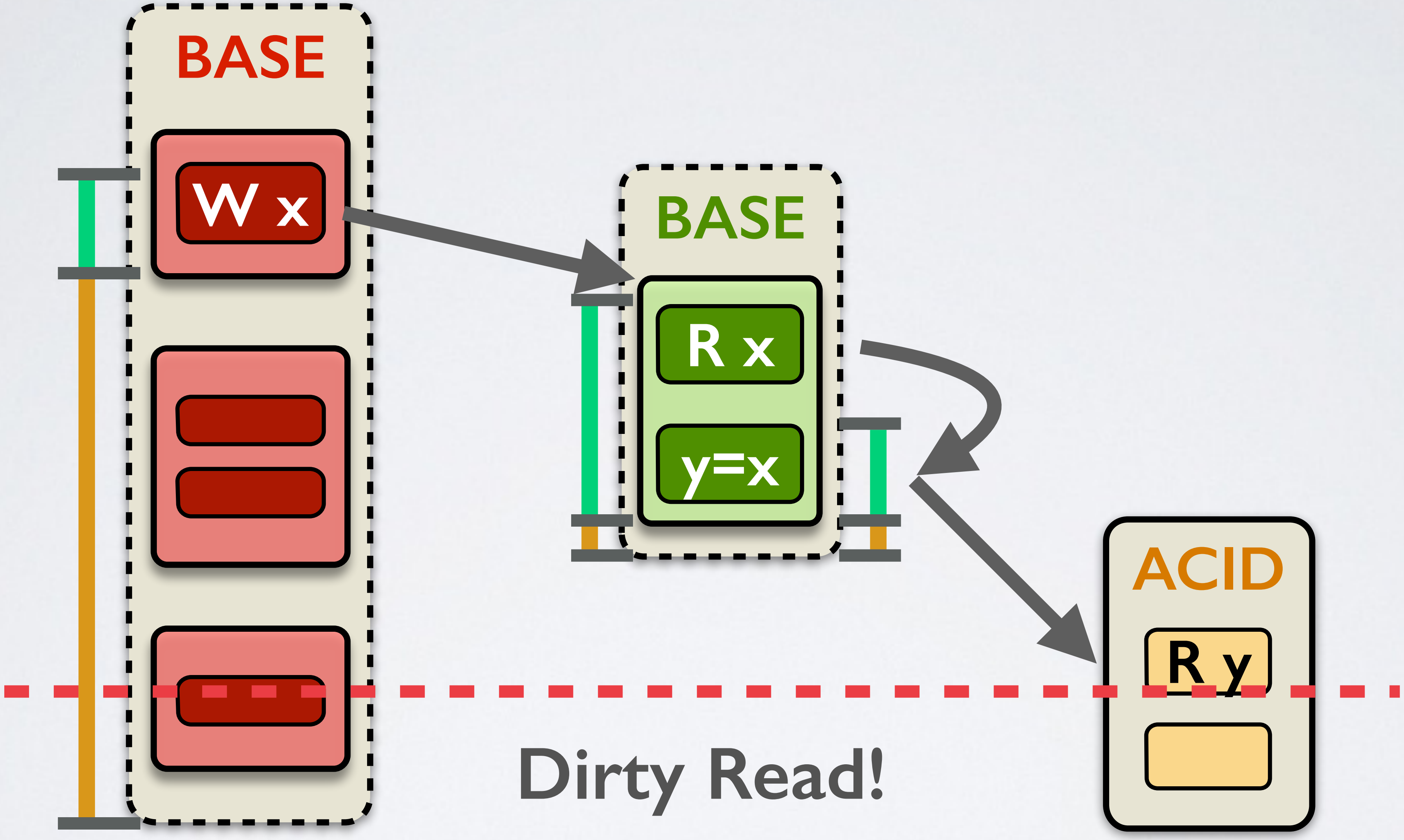
Wait

	AC-R	AC-W	alk-R	alk-W	sal-R	sal-W
AC-R	✓	✗	✓	✗	✓	✗
AC-W	✗	✗	✗	✗	✗	✗
alk-R	✓	✗	✓	✗	✓	✓
alk-W	✗	✗	✗	✗	✓	✓
sal-R	✓	✗	✓	✓	✓	✓
sal-W	✗	✗	✓	✓	✓	✓

Lock Table

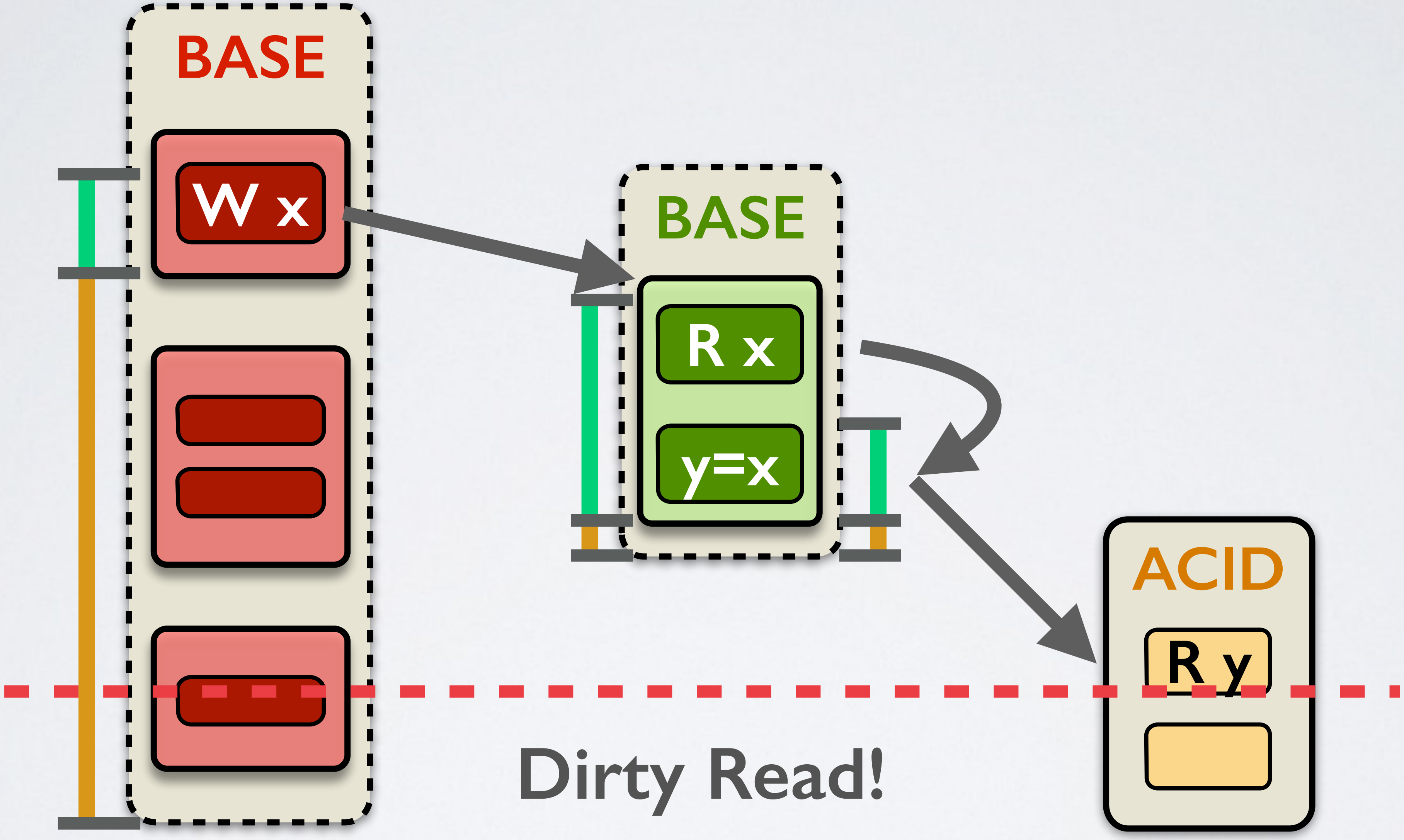
Conflict only with ACID locks

A SUBTLE PROBLEM



ACID reads uncommitted value of x!

A SUBTLE PROBLEM



For the solution, please read our paper

THE BOTTOM LINE

Guarantee

Salt prevents all ACID transactions from being affected by BASE transactions either directly or indirectly.

SALT

Motivation

Base Transactions & Salt Isolation

Achieving Salt Isolation

Evaluation

QUESTIONS TO ANSWER

What is the performance gain of Salt compared to ACID?

Can we get most performance gain compared to the BASE approach?

EXPERIMENTAL SETUP

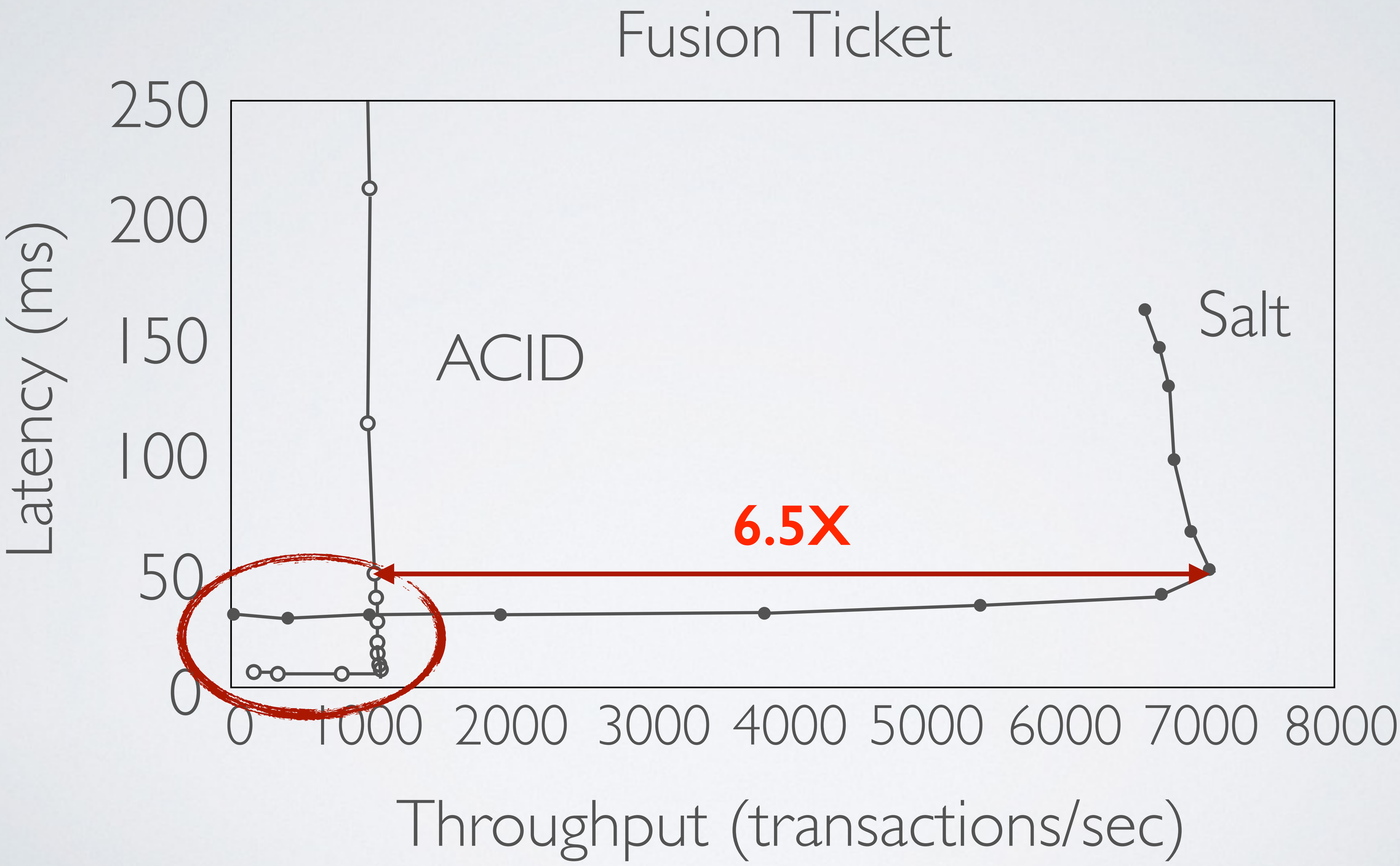
Configuration

- Emulab Cluster (Dell Power Edge R710)
- 10 shards, 3-way replicated

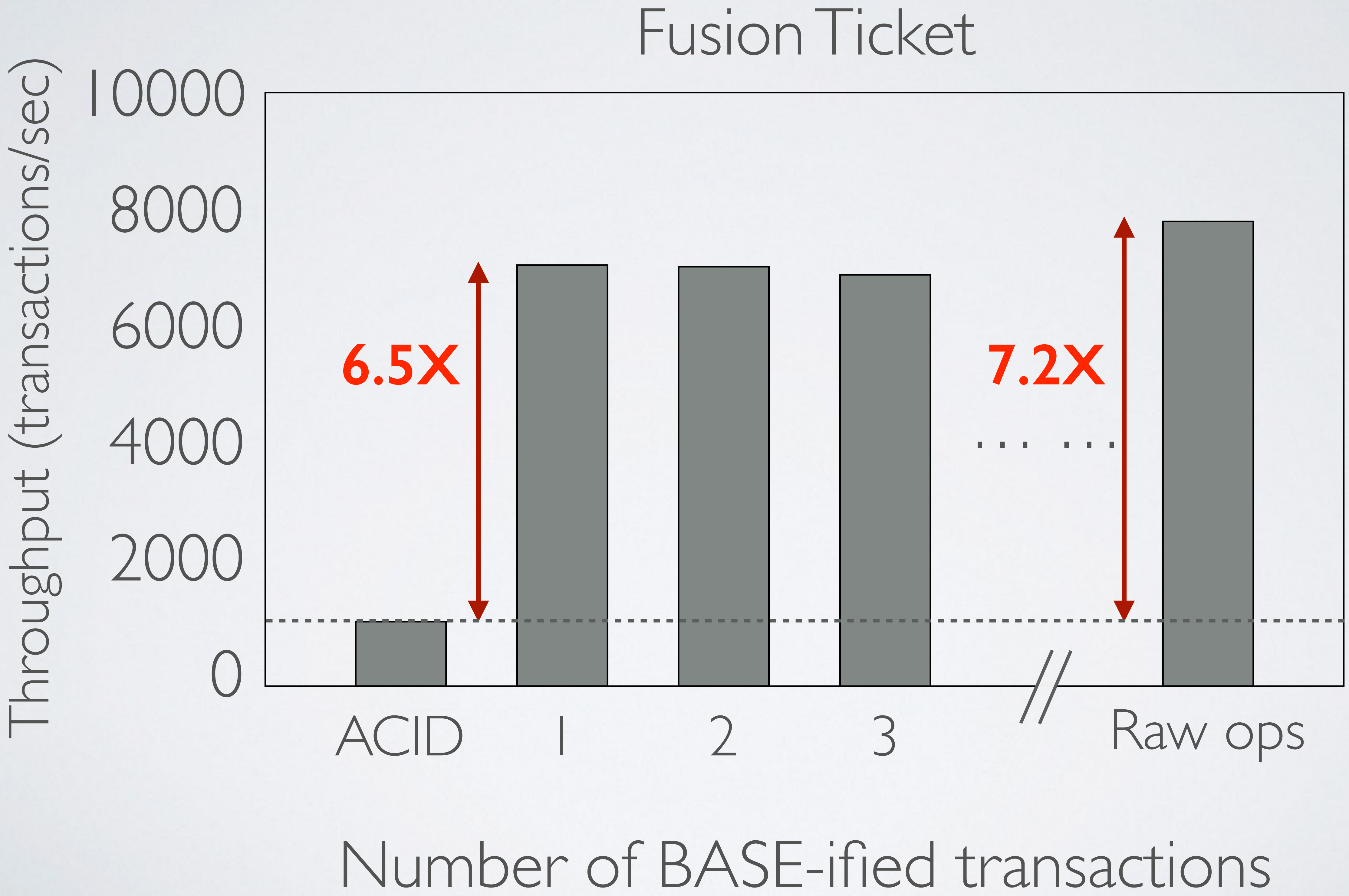
Workloads

- TPC-C
- Fusion Ticket
- Microbenchmarks

PERFORMANCE GAIN



REAP MOST PERFORMANCE OF BASE



RELATED WORK

Optimizing ACID Performance

- H-Store, Granola, F1, Sagas, Transaction Chain, Calvin …

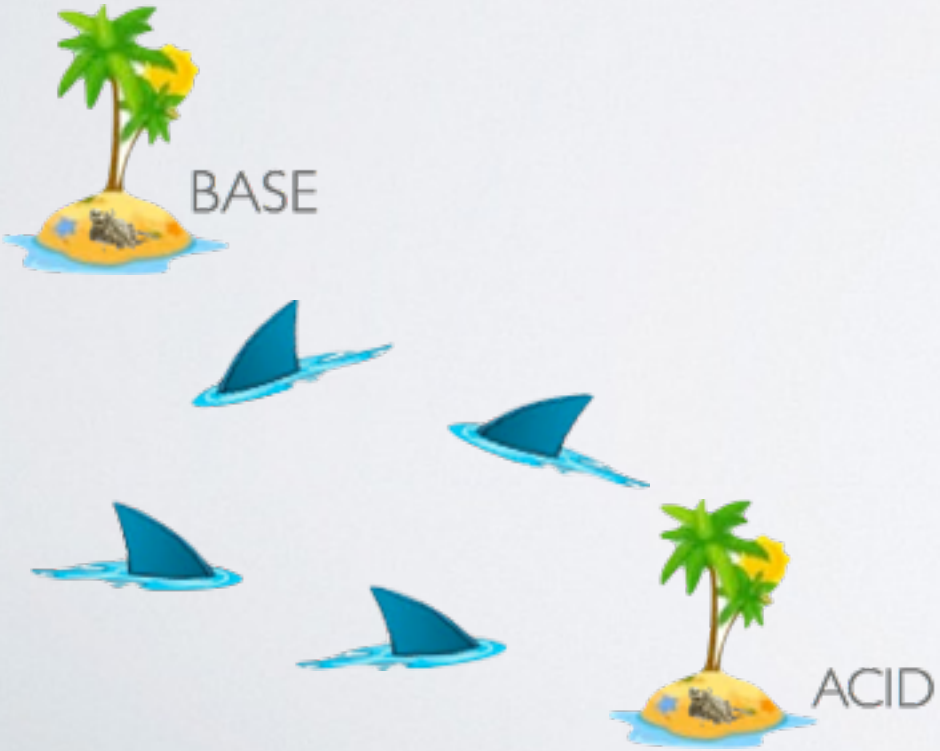
BASE with enhanced semantics (e.g., partition local transactions)

- ElasTras, G-Store, Megastore …

SALT

Pain Point

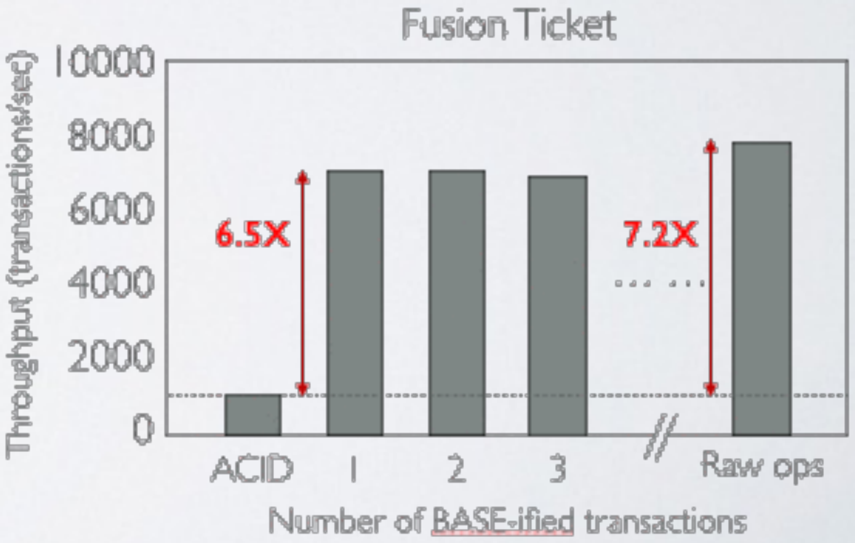
Transactional systems do not scale



Key Abstraction

Base Transaction

Promising Results


Fusion Ticket
Throughput (transactions/sec)
6.5X
7.2X
ACID
Raw ops
Number of BASE-ified transactions