SALT

Lorenzo Alvisi
The University of Texas at Austin
SO, YOU HAD A GOOD IDEA
SO, YOU HAD A GOOD IDEA

Time to start a company!
LOTS TO THINK ABOUT...
LOTS TO THINK ABOUT...

Understand your potential customers!

A business plan!

Funding!
LOTS TO THINK ABOUT...

Understand your potential customers!

A business plan!

Funding!
LOTS TO THINK ABOUT...

Understand your potential customers!

A business plan!

Funding!

Which database?
K.I.S.S.

A relational database
with **ACID** transactions

Easy to develop
Easy to reason about
THE FANTASTIC FOUR
THE FANTASTIC FOUR

Atomicity

Either all changes to the state happen, or none
THE FANTASTIC FOUR

Atomicity
Either all changes to the state happen, or none

Consistency
A transaction is a correct transformation of the state
THE FANTASTIC FOUR

**Atomicity**
Either all changes to the state happen, or none

**Consistency**
A transaction is a correct transformation of the state

**Isolation**
Regulates which states generated in executing T are visible to transactions executing concurrently to T
## THE FANTASTIC FOUR

<table>
<thead>
<tr>
<th>Atomicity</th>
<th>Either all changes to the state happen, or none</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consistency</td>
<td>A transaction is a correct transformation of the state</td>
</tr>
<tr>
<td>Isolation</td>
<td>Regulates which states generated in executing T are visible to transactions executing concurrently to T</td>
</tr>
<tr>
<td>Durability</td>
<td>Once a transaction commits, its state changes survive failures</td>
</tr>
</tbody>
</table>
DISASTER

You are wildly successful
THE ALTERNATIVE: BASE

- Basically
- Available,
- Soft state,
- Eventually consistent
THE ALTERNATIVE: BASE

Basically Available, Soft state, Eventually consistent

Application

Storage Interface
THE ALTERNATIVE: BASE

- Custom code for better performance

- Basically Available, Soft state, Eventually consistent

Application

BASE Storage (e.g. put, get)

Storage Interface
THE ALTERNATIVE: BASE

- Custom code for better performance

BASE Storage (e.g. put, get)

Application

Implement Consistency

Storage Interface

Basically Available, Soft state, Eventually consistent
THE ALTERNATIVE: BASE

- Custom code for better performance
- Complexity gets quickly out of control

BASE Storage (e.g. put, get)

Application

Implement Consistency

Storage Interface

Basically Available, Soft state, Eventually consistent
A STARK CHOICE

Performance

Ease of programming
A STARK CHOICE

Performance

Ease of programming

ACID
A STARK CHOICE

Performance

Ease of programming

BASE

ACID
A STARK CHOICE

Performance

Ease of programming

BASE

ACID
A STARK CHOICE

Performance

Ease of Programming & Performance

BASE

ACID

Ease of Programming & Performance

Shark Island
SALT
SALT
OUTLINE

An Opportunity
OUTLINE

An Opportunity

BASE Txs
OUTLINE

An Opportunity

BASE Txs

SALT Isolation
OUTLINE

An Opportunity

BASE Txs

SALT Isolation

Evaluation
Vilfredo Pareto
20% of the causes account for 80% of the effects

Vilfredo Pareto
NOT ALL TRANSACTIONS ARE CREATED EQUAL

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

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

Vilfredo Pareto
AN OPPORTUNITY
AN OPPORTUNITY

- Identify critical transactions
AN OPPORTUNITY

- Identify critical transactions
- BASE-ify only critical transactions without affecting others
MORE CONCURRENCY!
MORE CONCURRENCY!

Transfer

Is c ≥ $10?

\[ c = c - $10 \]

\[ s = s + $10 \]
Transfer

Is \( c \geq 10 \)?
\( c = c - 10 \)
\( s = s + 10 \)

Part 1

Is \( c \geq 10 \)?
\( c = c - 10 \)

Part 2

\( s = s + 10 \)

MORE CONCURRENCY!
MORE CONCURRENCY!

Transfer
Is c ≥ $10?
c = c - $10
s = s + $10

Transfer
Is c ≥ $10?
c = c - $10
s = s + $10

Transfer
Part 1
Is c ≥ $10?
c = c - $10

Transfer
Part 2
s = s + $10

Time
Transfer
Is $c \geq 10$?
c = c - $10$
s = s + $10$

Transfer
Part 1
Is $c \geq 10$?
c = c - $10$
Part 2
s = s + $10$

Transfer
Part 1
Is $c \geq 10$?
c = c - $10$
Part 2
s = s + $10$

Time
MORE CONCURRENCY!
MORE COMPLEXITY!
MORE COMPLEXITY!

Transfer

Is c≥$10?

\[ c = c - $10 \]

\[ s = s + $10 \]
MORE COMPLEXITY!

Balance
- Read c
- Read s

Transfer
- Is $c \geq 10$?
- $c = c - 10$
- $s = s + 10$
MORE COMPLEXITY!

Balance
- Read c
- Read s

Transfer
- Is c ≥ $10?
- c = c - $10
- s = s + $10
Is $c \geq 10$?

- $c = c - 10$
- $s = s + 10$

Balance
- Read $c$
- Read $s$

Transfer
More Complexity!

Balance
- Read c
- Read s

Transfer

Part 1
- Is \( c \geq 10 \)?
- \( c = c - 10 \)

Part 2
- \( s = s + 10 \)
MORE COMPLEXITY!

**Part 1**
- Is c ≥ $10?
- c = c - $10

**Part 2**
- s = s + $10

**Balance**
- Read c
- Read s

Finer Isolation for one transaction may affect all transactions!!
Performance vs Complexity
Performance vs Complexity

Better Performance
Performance vs Complexity

Better Performance

More Interleavings
Performance vs Complexity

Better Performance

More Interleaveings

Greater Complexity
Performance vs Complexity

More Interleavings
Performance vs Complexity

More Interleavings selectively
Performance vs Complexity

More Interleavings selectively
Balance

Read c

Read s

Transfer 1

Is c ≥ $10?

c = c - $10

s = s + $10

Transfer 2

Is c ≥ $10?

c = c - $10

s = s + $10
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
Balance
Read c
Read s

Transfer 1
Is $c \geq 10$?
$c = c - 10$
$s = s + 10$

Transfer 2
Is $c \geq 10$?
$c = c - 10$

$s = s + 10$
Decouple
Nested Transactions

decouple Atomicity and Isolation
Nested Transactions

decouple Atomicity and Isolation
for finer-grained Atomicity
Nested Transactions

decouple Atomicity and Isolation
for finer-grained Atomicity
Acid txn
BASE TRANSACTION

BASE transaction

alkaline txn

alkaline txn

Atomicity
Durability
BASE TRANSACTION

Different Isolation guarantees for different types of transactions
BASE WITH BASE

Fine Isolation granularity between BASE transactions
BASE WITH BASE

Fine Isolation granularity between BASE transactions
BASE WITH ACID

Coarse Isolation granularity to ACID transactions
BASE WITH ACID

Coarse Isolation granularity to ACID transactions
SALT ISOLATION
SALT ISOLATION
SALT ISOLATION

To BASE transactions:
a sequence of small
ACID transactions
SALT ISOLATION

To BASE transactions: a sequence of small ACID transactions

To ACID transactions: a single, monolithic ACID transaction
SALT ISOLATION

To BASE transactions: a sequence of small ACID transactions

To ACID transactions: a single, monolithic ACID transaction

Performance & Ease of Programming
OUTLINE
OUTLINE

SALT Isolation
FLAVORS OF ISOLATION

It-That-Shall-Not-Be-Named
dirty writes - transaction modifies item previously modified by undecided transaction

Read-Uncommitted
dirty reads: one transaction may see uncommitted state of another transaction

Read-Committed
no dirty reads or writes, but allows for non-repeatable reads

Repeateable Reads
non repeatable range reads

Serializable
no bad phenomena - as if transactions were executed in some sequential order
FLAVORS OF ISOLATION

Read-Committed

no dirty reads or writes, but allows for non-repeatable reads
FLAVORS OF ISOLATION

Read-Committed

no dirty reads or writes, but allows for non-repeatable reads
FLAVORS OF ISOLATION

Read-Committed
no dirty reads or writes, but allows for non-repeatable reads

If

W1[x]

If

R2[x]

W2[x]
FLAVORS OF ISOLATION

Read-Committed
no dirty reads or writes, but allows for non-repeatable reads

If
\[ W1[x] \] completes before \[ W2[x] \]
\[ R2[x] \] starts
FLAVORS OF ISOLATION

Read-Committed
no dirty reads or writes, but allows for non-repeatable reads

If
W1[x] completes before R2[x] starts
then
W2[x]
FLAVORS OF ISOLATION

Read-Committed

no dirty reads or writes, but allows for non-repeatable reads

If

$W_1[x]$ completes before $R_2[x]$ starts

then

$T_1$ must decide before $R_2[x]$ starts

$W_2[x]$ starts
If $W_1[x]$ completes before $R_2[x]$ starts, then $T_1$ must decide before $R_2[x]$ starts.
If $o_1$ completes before $W_2[x]$ starts, then $T_1$ must decide before $R_2[x]$ starts.
If $O_1$ completes before $O_2$ starts then $T_1$ must decide before $R2[x]$ starts $W2[x]$ starts.
If $O_1$ completes before $O_2$ starts

then

$T_1$ must decide before $O_2$ starts
If \( L \cup O_1 \) completes before \( O_2 \) starts, then \( T_1 \) must decide before \( O_2 \) starts.
\[ L \cup O_1 \text{ completes before } S \cup O_2 \text{ starts} \]

\[ T_1 \text{ then must decide before } O_2 \text{ starts} \]
Isolation Property

Given

- $o_1$ of type $L$ in $T_1$, $o_2$ of type $S$ in $T_2$
- $o_1$ conflicting with $o_2$

If $o_1$ completes before $o_2$ starts, then $T_1$ must decide before $o_2$ starts
E PLURIBUS UNUM

Isolation Property

Given

- $o_1$ of type $L$ in $T_1$, $o_2$ of type $S$ in $T_2$
- $o_1$ conflicting with $o_2$

If $o_1$ completes before $o_2$ starts, then $T_1$ must decide before $o_2$ starts

<table>
<thead>
<tr>
<th>Isolation Level</th>
<th>L</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read-uncommitted (no dirty writes)</td>
<td>W</td>
<td>W</td>
</tr>
<tr>
<td>Read-committed (no dirty reads/writes)</td>
<td>W</td>
<td>R, W</td>
</tr>
<tr>
<td>Repeatable-read</td>
<td>R, W</td>
<td>R, W</td>
</tr>
<tr>
<td>Serializable</td>
<td>R, RR, W</td>
<td>R, RR, W</td>
</tr>
</tbody>
</table>
SALT ISOLATION

The Isolation Property holds as long as $T_1$ and $T_2$ are not both BASE transactions.
SALT ISOLATION

The Isolation Property holds as long as $T_1$ and $T_2$ are not both BASE transactions.

ACID
Isolated from all other transactions

Alkaline
Isolated from all ACID and alkaline (sub)transactions

BASE
Expose selected intermediate states to other BASE transactions
ONE MECHANISM

LOCKS
ONE MECHANISM

LOCKS

Three flavors

ACID locks
Alkaline locks
Saline locks
ACID LOCKS

ACID 1

Write x
ACID LOCKS

ACID I

Write x

ACID 2

Read x

Wait
ACID LOCKS

ACID 1
Write x
Read x
Execute

ACID 2
ACID LOCKS

ACID 1
- Write x

ACID 2
- Read x

Execute

Lock Table

<table>
<thead>
<tr>
<th></th>
<th>AC-R</th>
<th>AC-W</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC-R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AC-W</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ACID LOCKS

ACID 1
- Write
- Execute

ACID 2
- Read
- Execute

Lock Table

<table>
<thead>
<tr>
<th></th>
<th>AC-R</th>
<th>AC-W</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC-R</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>AC-W</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ACID LOCKS

ACID 1
Write x

ACID 2
Read x

Lock Table

<table>
<thead>
<tr>
<th></th>
<th>AC-R</th>
<th>AC-W</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC-R</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>AC-W</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>
LOCKS?
LOCKS?

BASE

Rx

Wait

BASE

Wx

Wait
LOCKS?
LOCKS?

ACID

Rx

BASE

Wx

RX

Execute

Execute
ALKALINE LOCKS
ALKALINE LOCKS

alkaline lock

ACID
Rx

BASE
Wx

Wait

BASE
Rx

Wait
ALKALINE LOCKS

alkaline lock

Wait

Wait

ACID
Rx

BASE
Wx

BASE
Rx

Lock Table

<table>
<thead>
<tr>
<th></th>
<th>AC-R</th>
<th>AC-W</th>
<th>alk-R</th>
<th>alk-W</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC-R</td>
<td>✓</td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AC-W</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>alk-R</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>alk-W</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ALKALINE LOCKS

alkaline lock

ACID
Rx

Wait

BASE
Wx

Wait

BASE
Rx

Lock Table

<table>
<thead>
<tr>
<th></th>
<th>AC-R</th>
<th>AC-W</th>
<th>alk-R</th>
<th>alk-W</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC-R</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>AC-W</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td>alk-R</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>alk-W</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
ALKALINE LOCKS

alkaline lock

Wait

Wait

Lock Table

<table>
<thead>
<tr>
<th></th>
<th>AC-R</th>
<th>AC-W</th>
<th>alk-R</th>
<th>alk-W</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC-R</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>AC-W</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>alk-R</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>alk-W</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>
SALINE LOCKS

Lock Table

<table>
<thead>
<tr>
<th></th>
<th>AC-R</th>
<th>AC-W</th>
<th>alk-R</th>
<th>alk-W</th>
<th>sal-R</th>
<th>sal-W</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC-R</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AC-W</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td></td>
<td></td>
</tr>
<tr>
<td>alk-R</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td></td>
<td></td>
</tr>
<tr>
<td>alk-W</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sal-R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sal-W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## SALINE LOCKS

### Lock Table

<table>
<thead>
<tr>
<th></th>
<th>AC-R</th>
<th>AC-W</th>
<th>alk-R</th>
<th>alk-W</th>
<th>sal-R</th>
<th>sal-W</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC-R</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AC-W</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td></td>
<td></td>
</tr>
<tr>
<td>alk-R</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td></td>
<td></td>
</tr>
<tr>
<td>alk-W</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sal-R</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sal-W</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SALINE LOCKS

- **saline lock**

<table>
<thead>
<tr>
<th></th>
<th>AC-R</th>
<th>AC-W</th>
<th>alk-R</th>
<th>alk-W</th>
<th>sal-R</th>
<th>sal-W</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC-R</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AC-W</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td></td>
<td></td>
</tr>
<tr>
<td>alk-R</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td></td>
<td></td>
</tr>
<tr>
<td>alk-W</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sal-R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>sal-W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>
SALINE LOCKS

<table>
<thead>
<tr>
<th></th>
<th>AC-R</th>
<th>AC-W</th>
<th>alk-R</th>
<th>alk-W</th>
<th>sal-R</th>
<th>sal-W</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC-R</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>AC-W</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>alk-R</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td></td>
<td></td>
</tr>
<tr>
<td>alk-W</td>
<td>✗</td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sal-R</td>
<td>✓</td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sal-W</td>
<td>✗</td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SALINE LOCKS

<table>
<thead>
<tr>
<th></th>
<th>AC-R</th>
<th>AC-W</th>
<th>alk-R</th>
<th>alk-W</th>
<th>sal-R</th>
<th>sal-W</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC-R</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>AC-W</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>alk-R</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>alk-W</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>sal-R</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>sal-W</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
A SUBTLE PROBLEM

ACID can read uncommitted values!
A SUBTLE PROBLEM

ACID can read uncommitted values!
A SUBTLE PROBLEM

ACID can read uncommitted values!
A SUBTLE PROBLEM

ACID can read uncommitted values!
A SUBTLE PROBLEM

ACID can read uncommitted values!

Dirty Read!
THE BOTTOM LINE
THE BOTTOM LINE

THEOREM

Given isolation level A, Salt Isolation protects all ACID transaction (both directly and indirectly) from all undesirable phenomena prevented by A
OUTLINE

Evaluation
QUESTIONS

What is the performance gain of Salt compared to ACID?

How much effort is required to gain performance comparable to BASE?
EXPERIMENTAL SETUP

Configuration

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

Workloads

- TPC-C
- Fusion Ticket
- Microbenchmarks
PERFORMANCE GAIN

Fusion Ticket

Latency (ms)

Throughput (transactions/sec)

ACID

Salt

6.5x

Latency for Fusion Ticket:

- ACID: 50ms
- Salt: 600ms

6.5x performance gain for Salt over ACID.
PROGRAMMING EFFORT VS PERFORMANCE

Throughput (transactions/sec)

Number of BASE-ified transactions

Fusion Ticket

ACID  1  2  3

Raw ops

…...
PROGRAMMING EFFORT VS PERFORMANCE

Throughput (transactions/sec)

<table>
<thead>
<tr>
<th>Number of BASE-ified transactions</th>
<th>Fusion Ticket</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACID</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>Raw ops</td>
</tr>
</tbody>
</table>

Throughput comparison:

- ACID: 0 transactions/sec
- Fusion Ticket: 6.5x improvement

Graph shows a significant increase in throughput for Fusion Ticket compared to ACID.
PROGRAMMING EFFORT VS PERFORMANCE

<table>
<thead>
<tr>
<th>Number of BASE-ified transactions</th>
<th>ACID</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>Raw ops</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput (transactions/sec)</td>
<td>0</td>
<td>6.5x</td>
<td>6.5x</td>
<td>6.5x</td>
<td>6.5x</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7.2x</td>
<td>7.2x</td>
<td>7.2x</td>
<td>7.2x</td>
</tr>
</tbody>
</table>

Fusion Ticket
Pain Point
Transactional systems do not scale

Key Abstraction
Base Transaction

Promising Results

<table>
<thead>
<tr>
<th>Number of BASE-fied transactions</th>
<th>Throughput (transactions/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACID</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1000</td>
</tr>
<tr>
<td>2</td>
<td>2000</td>
</tr>
<tr>
<td>3</td>
<td>3000</td>
</tr>
<tr>
<td>Raw ops</td>
<td>4000/5000</td>
</tr>
</tbody>
</table>

Fusion Ticket

6.5x
7.2x
SALT

Pain Point
Transactional systems do not scale

Key Abstraction
Base Transaction

Promising Results

![Graph showing throughput improvement with fusion ticket. The graph compares ACID and BASE-fied transactions, demonstrating a 6.5x and 7.2x increase in throughput for certain BASE-fied transactions compared to ACID.](image)