

THIS PRESENTATION HAS BEEN RATED

NC-PC

**SIGMOD 2014 PC MEMBERS
STRONGLY CAUTIONED**



Some material may compromise double-blind reviewing

THE FUTURE OF HUMANITY MAY BE AT STAKE

www.filmratings.com

www.mpaa.org



WORLD WAR

HPTS

An oral history of the
zombie pandemic
(1997-2018)

Peter Bailis
Sept. 23, 2023



**1976-1997: ACID
is a roaring success**

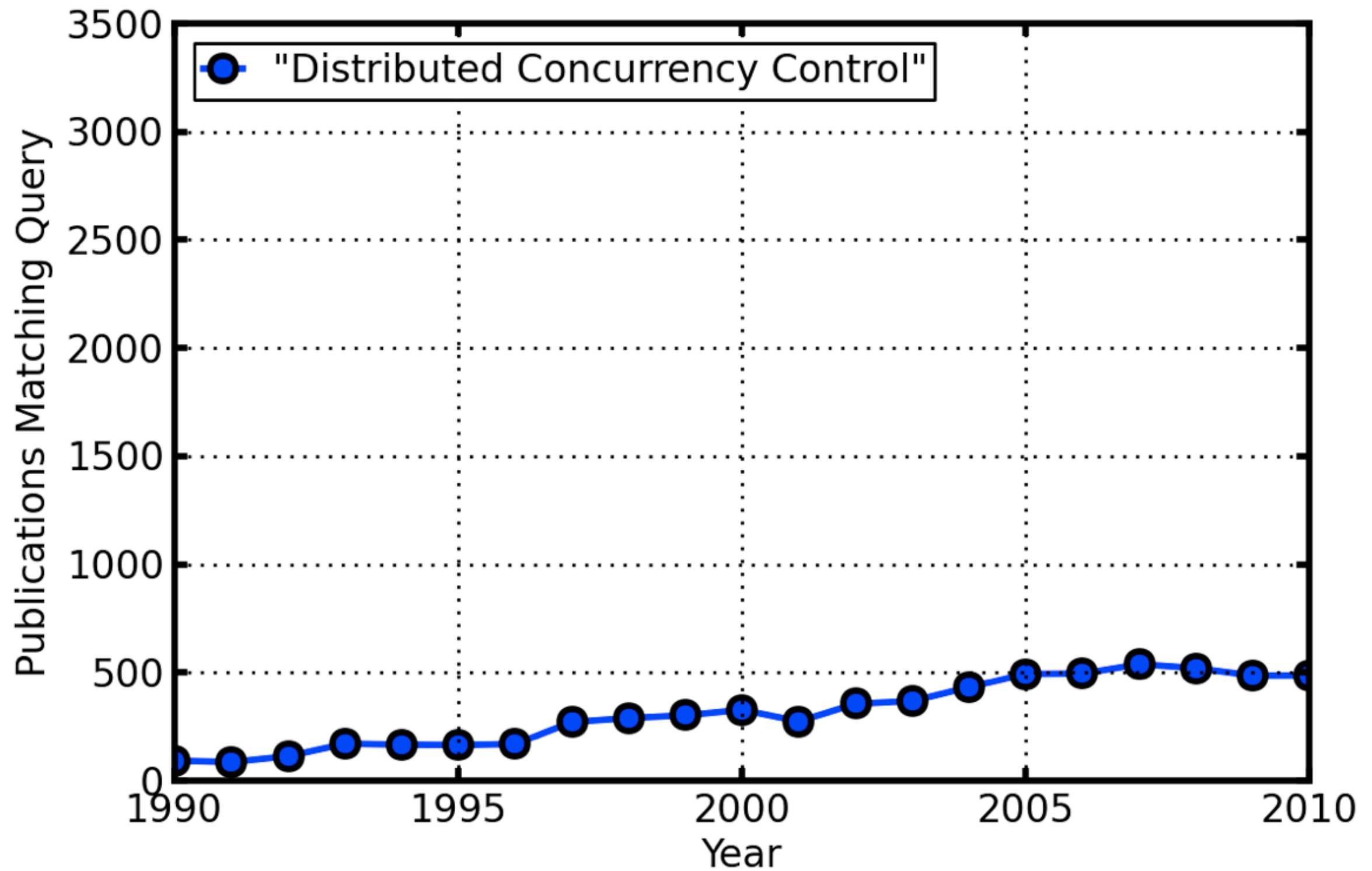


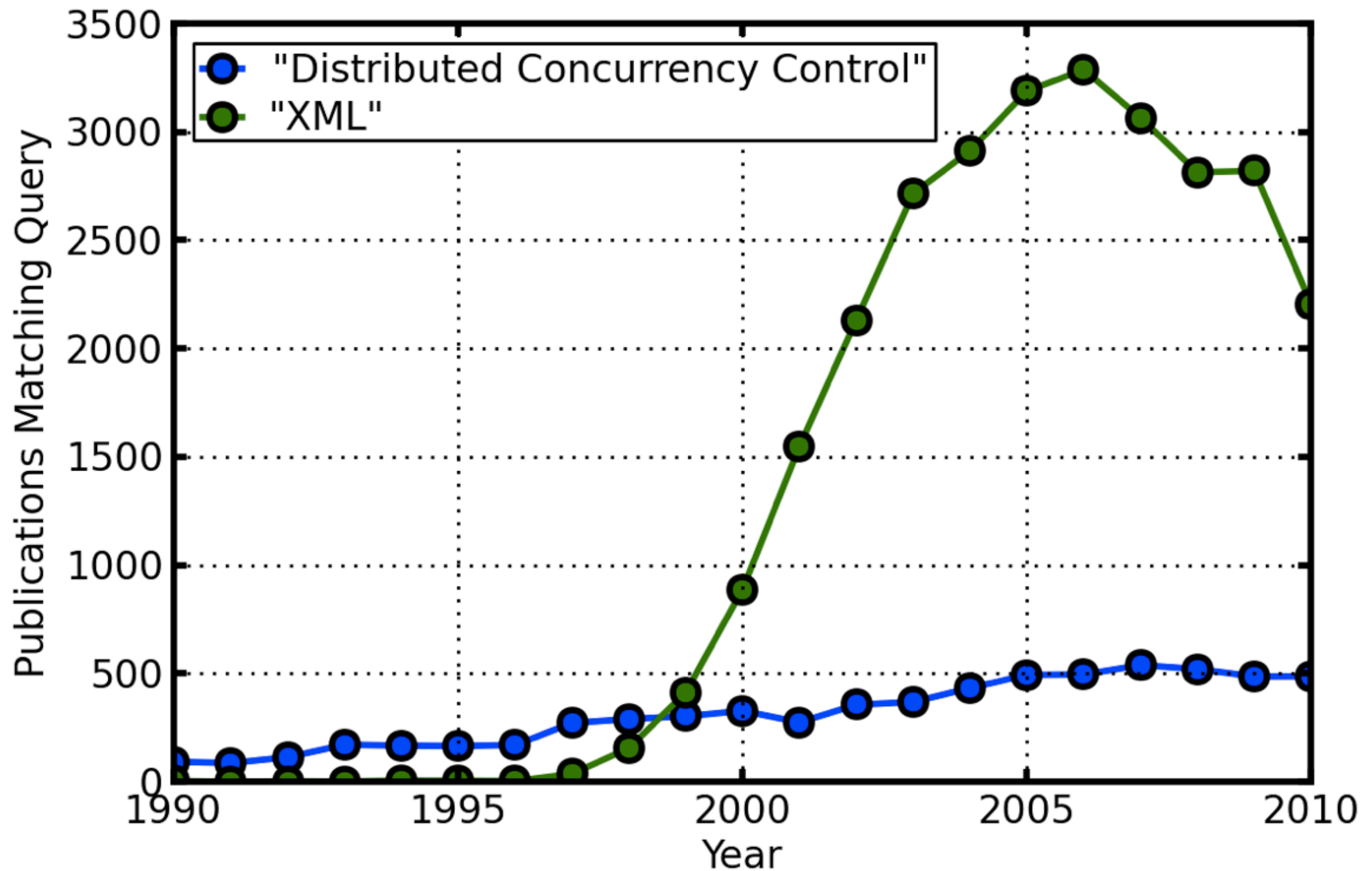
**1976-1997: ACID
is a roaring success**



**SERIALIZABLE
TRANSACTIONS**









 1997



1997



1998-2006



1997



1998-2

Patient Number: 19204920

Age: 32 years, 11 days

Time since infection: 3 months

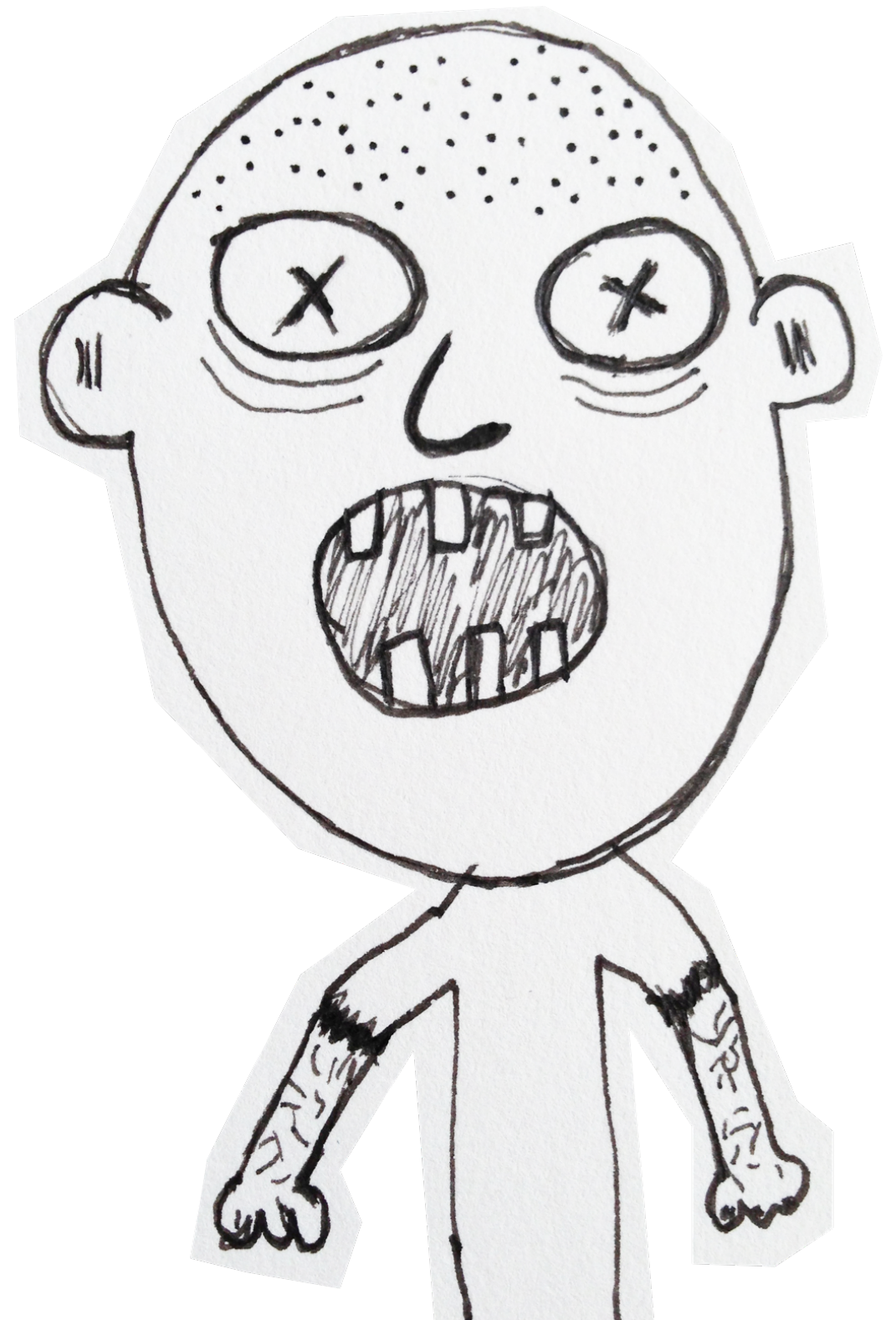


SYMPTOMS

Patient Number: 19204920

Age: 32 years, 11 days

Time since infection: 3 months

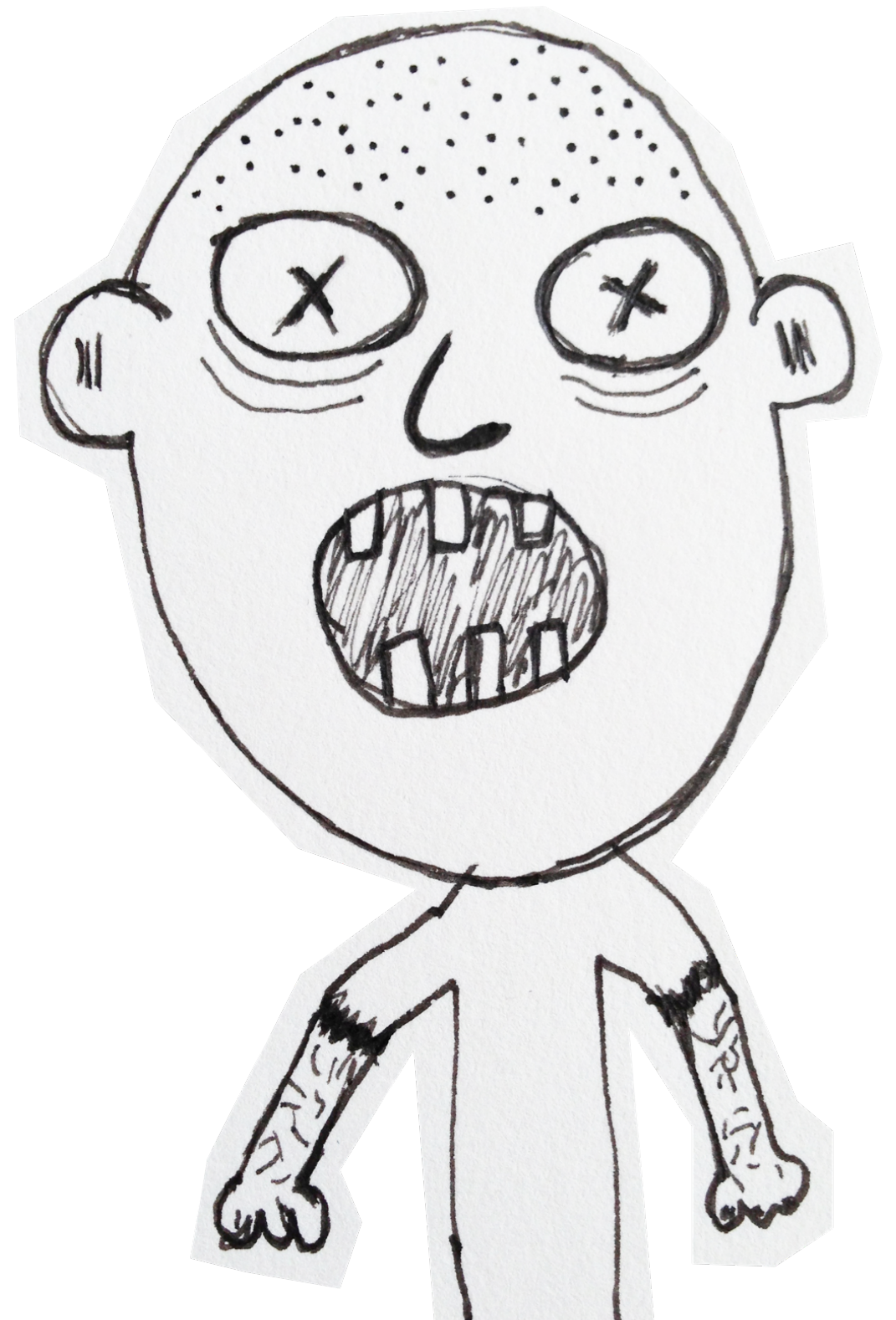


SYMPTOMS

Patient Number: 19204920
Age: 32 years, 11 days
Time since infection: 3 months

Complains about

- ACID scalability,
- high availability,
- scale-out,
- partitioning



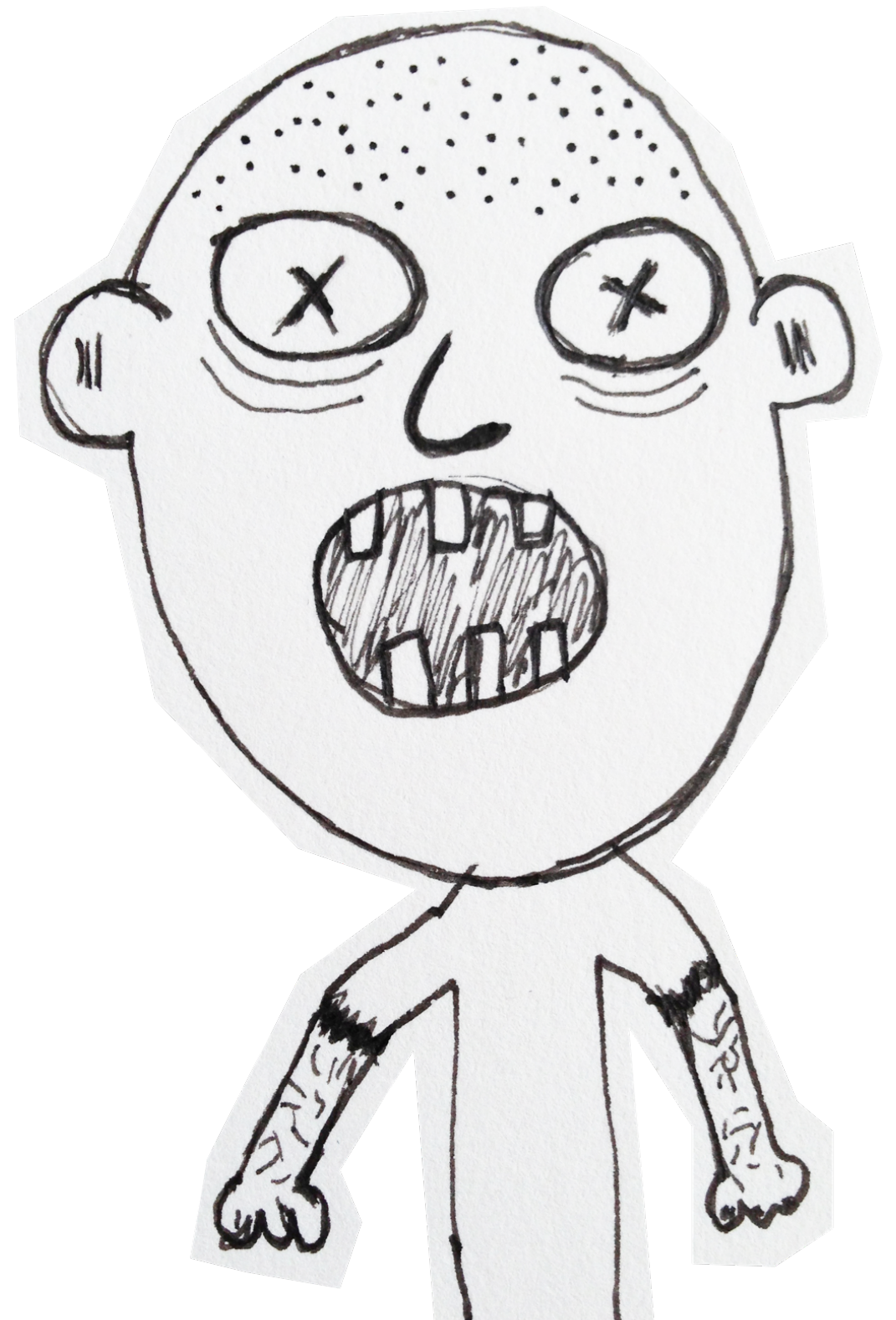
SYMPTOMS

Patient Number: 19204920
Age: 32 years, 11 days
Time since infection: 3 months

Complains about

- ACID scalability,
- high availability,
- scale-out,
- partitioning

Obsession with
“web scale”



SYMPTOMS

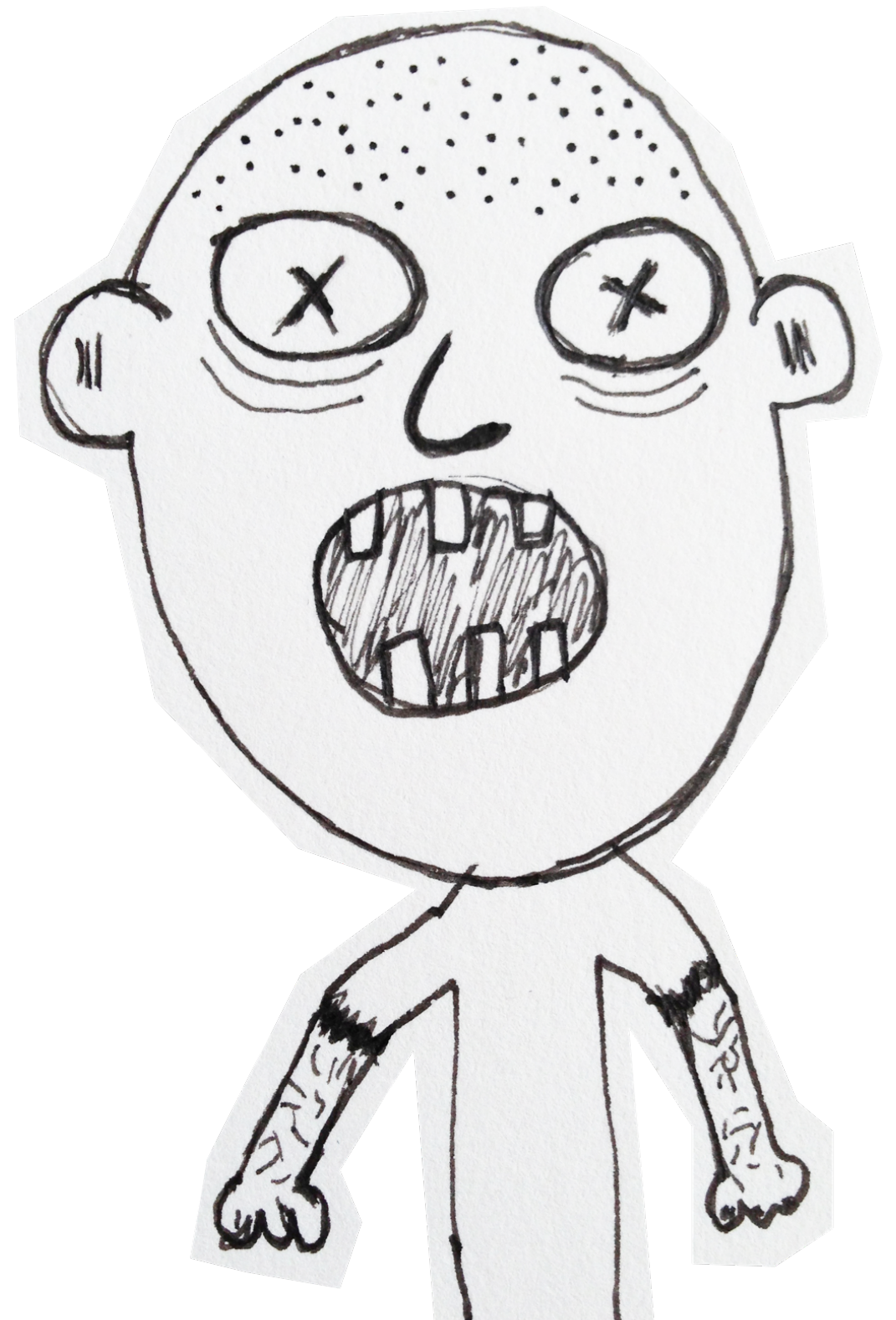
Patient Number: 19204920
Age: 32 years, 11 days
Time since infection: 3 months

Complains about

- ACID scalability,
- high availability,
- scale-out,
- partitioning

Obsession with
“web scale”

Strong case of
“Not Invented Here”

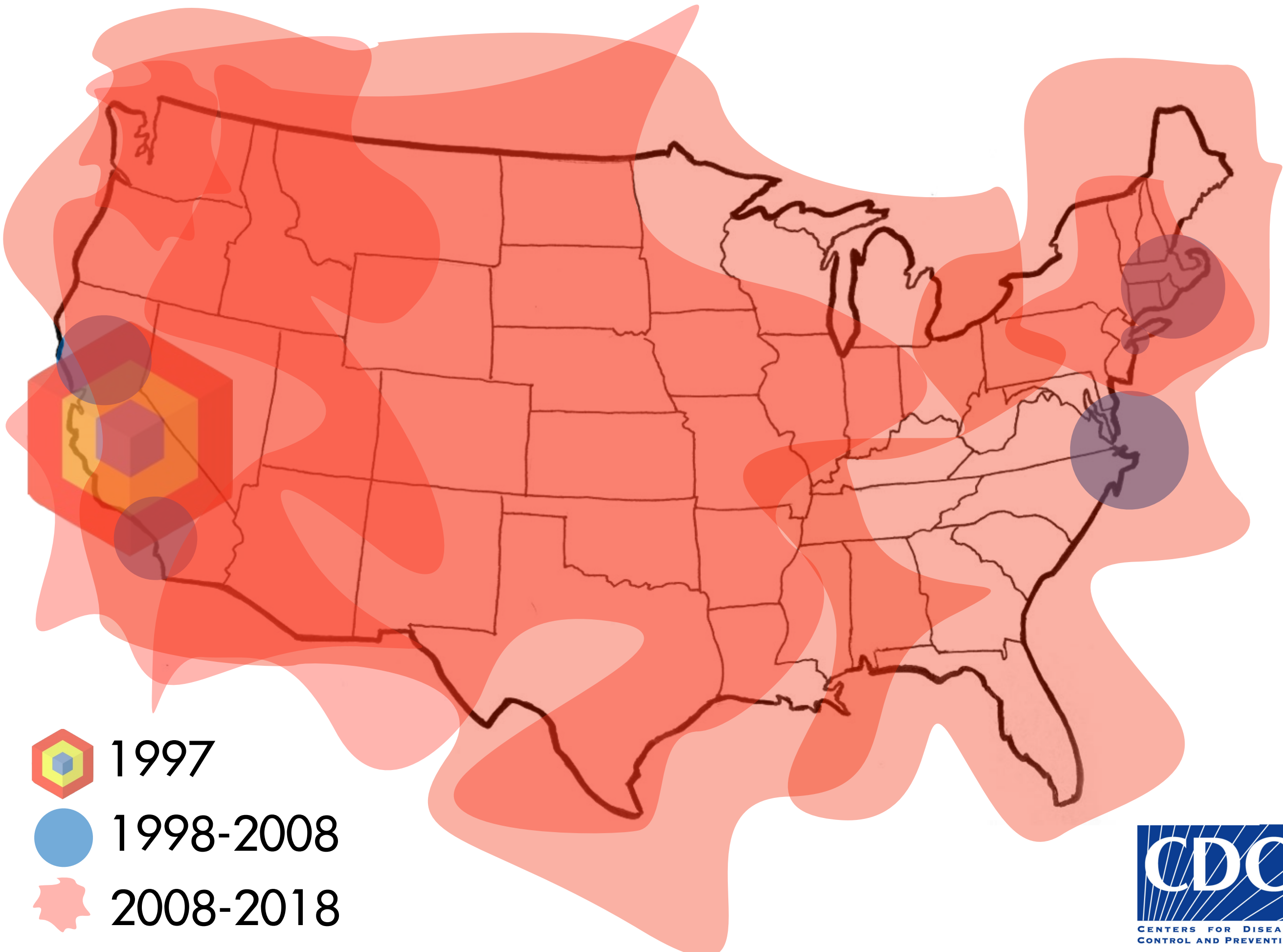







1997



1998-2008



-  1997
-  1998-2008
-  2008-2018



1997



1998-2007



2008-2017



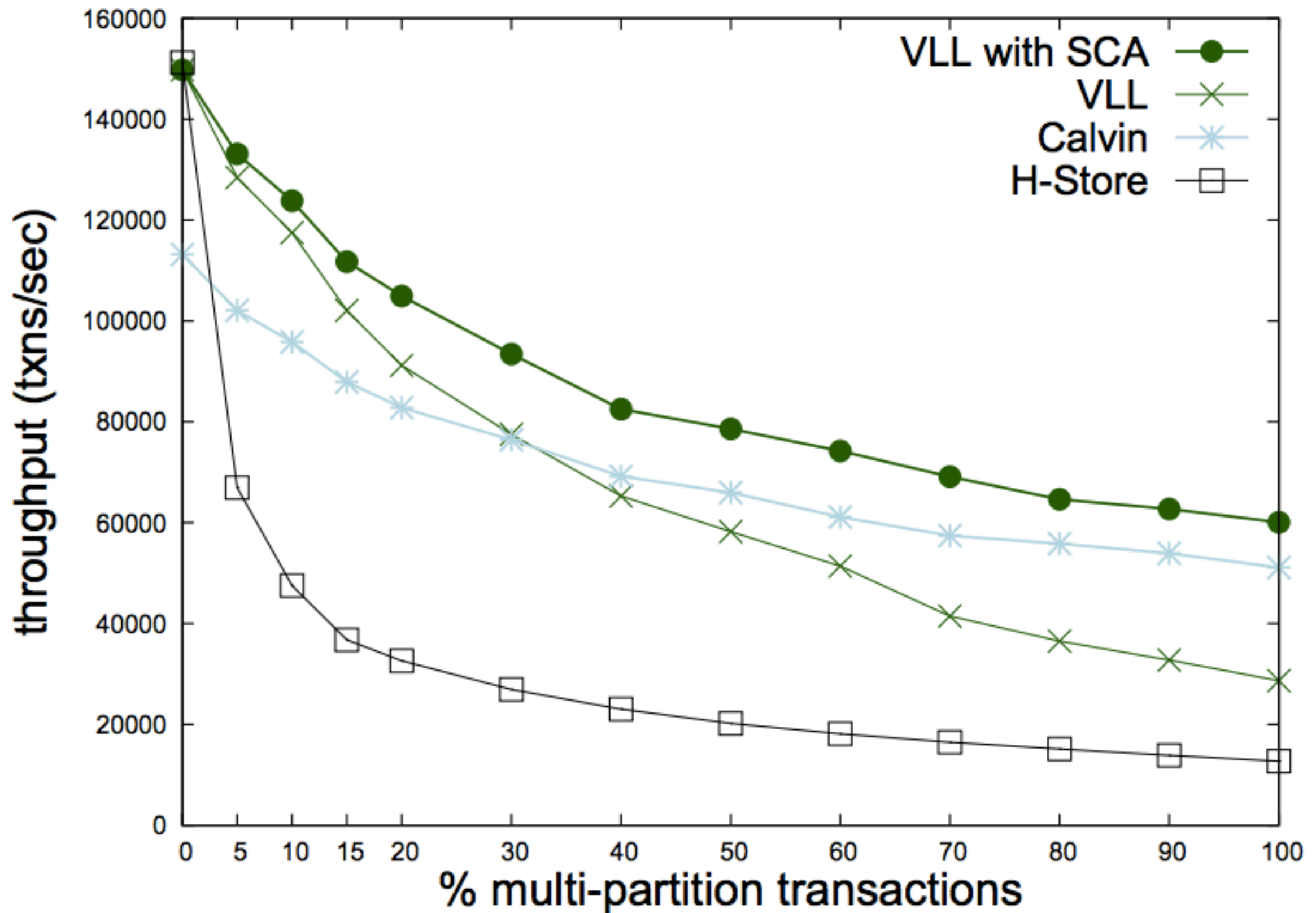
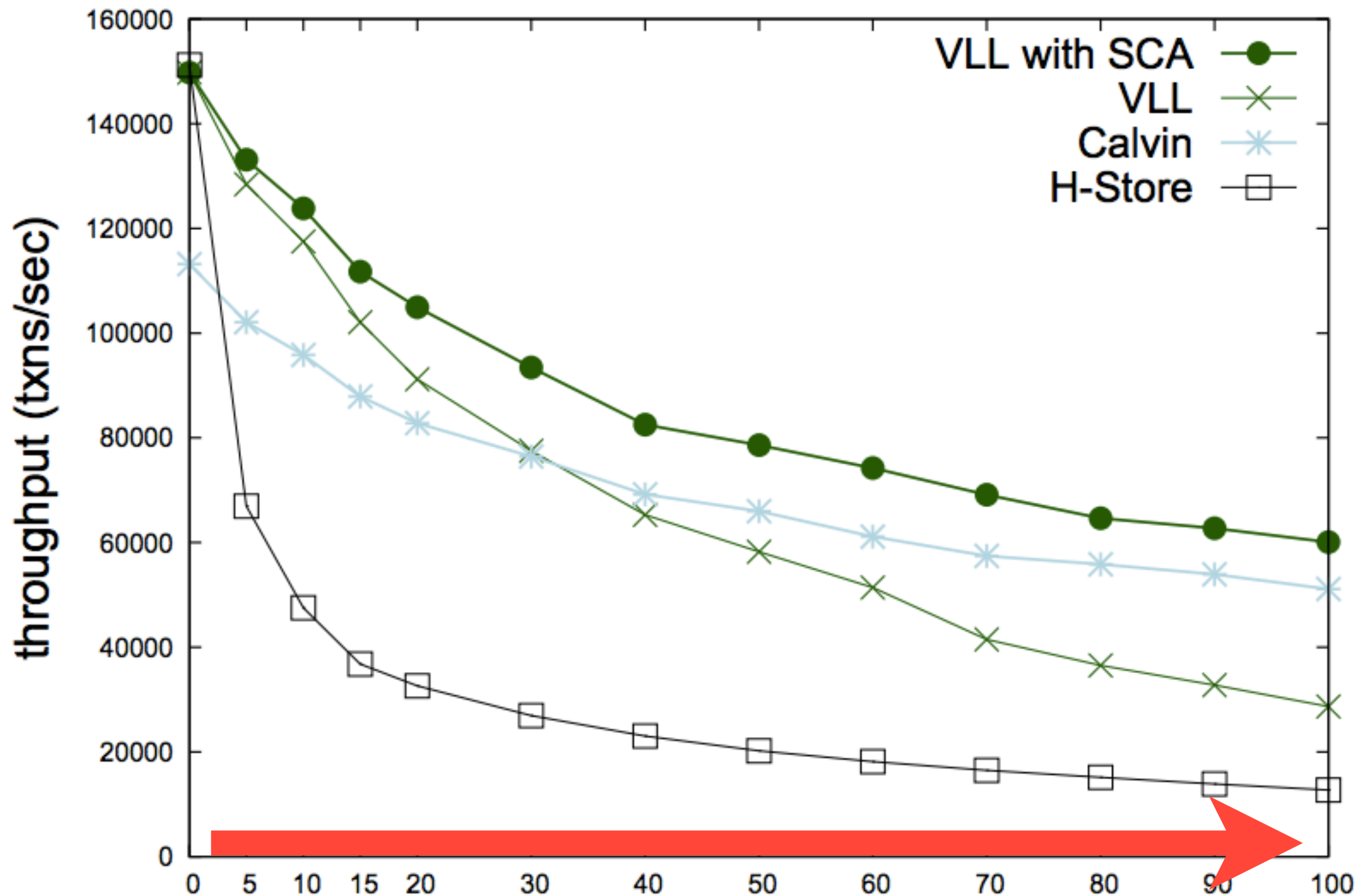
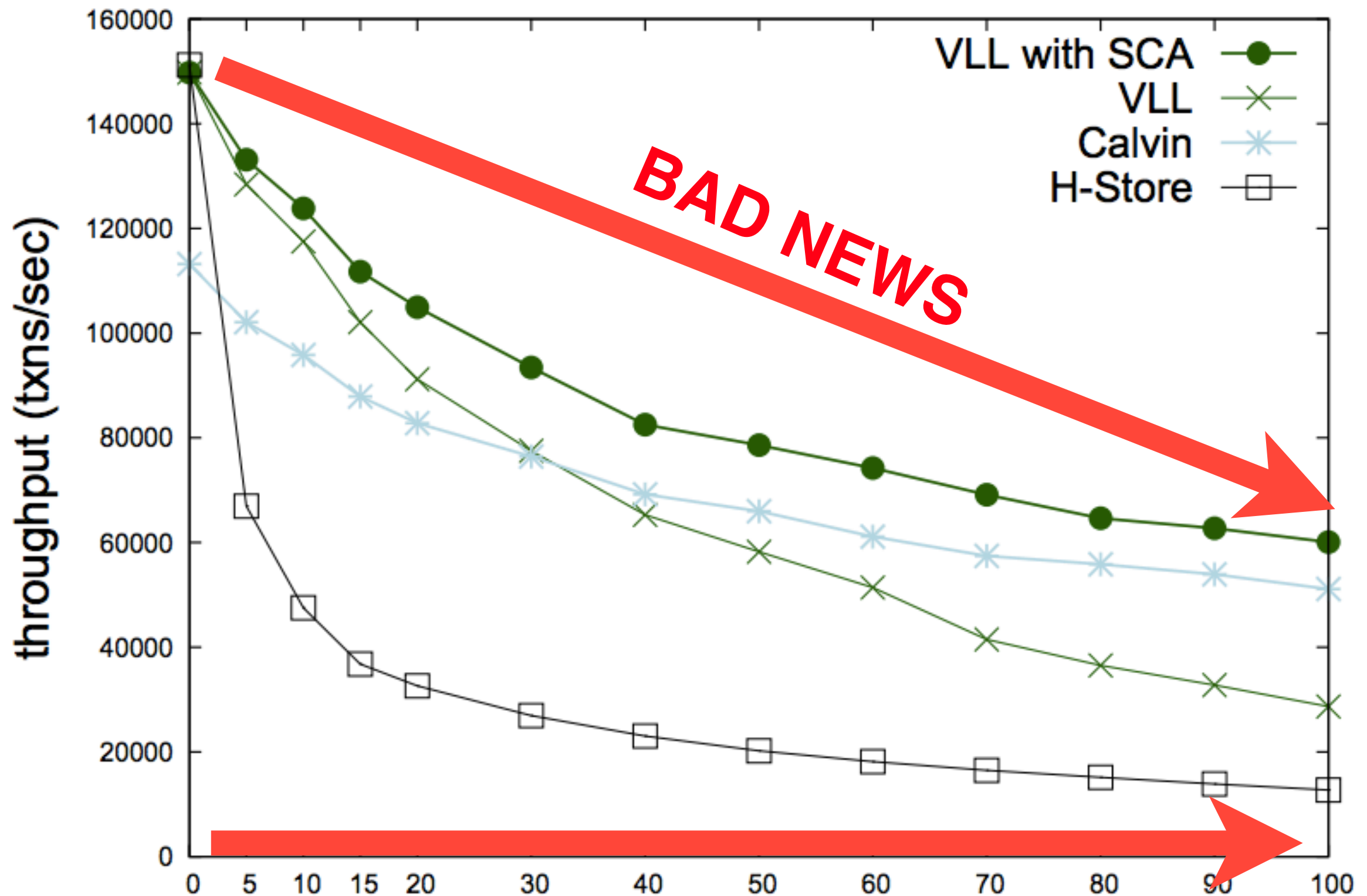


Figure 9: TPC-C throughput.



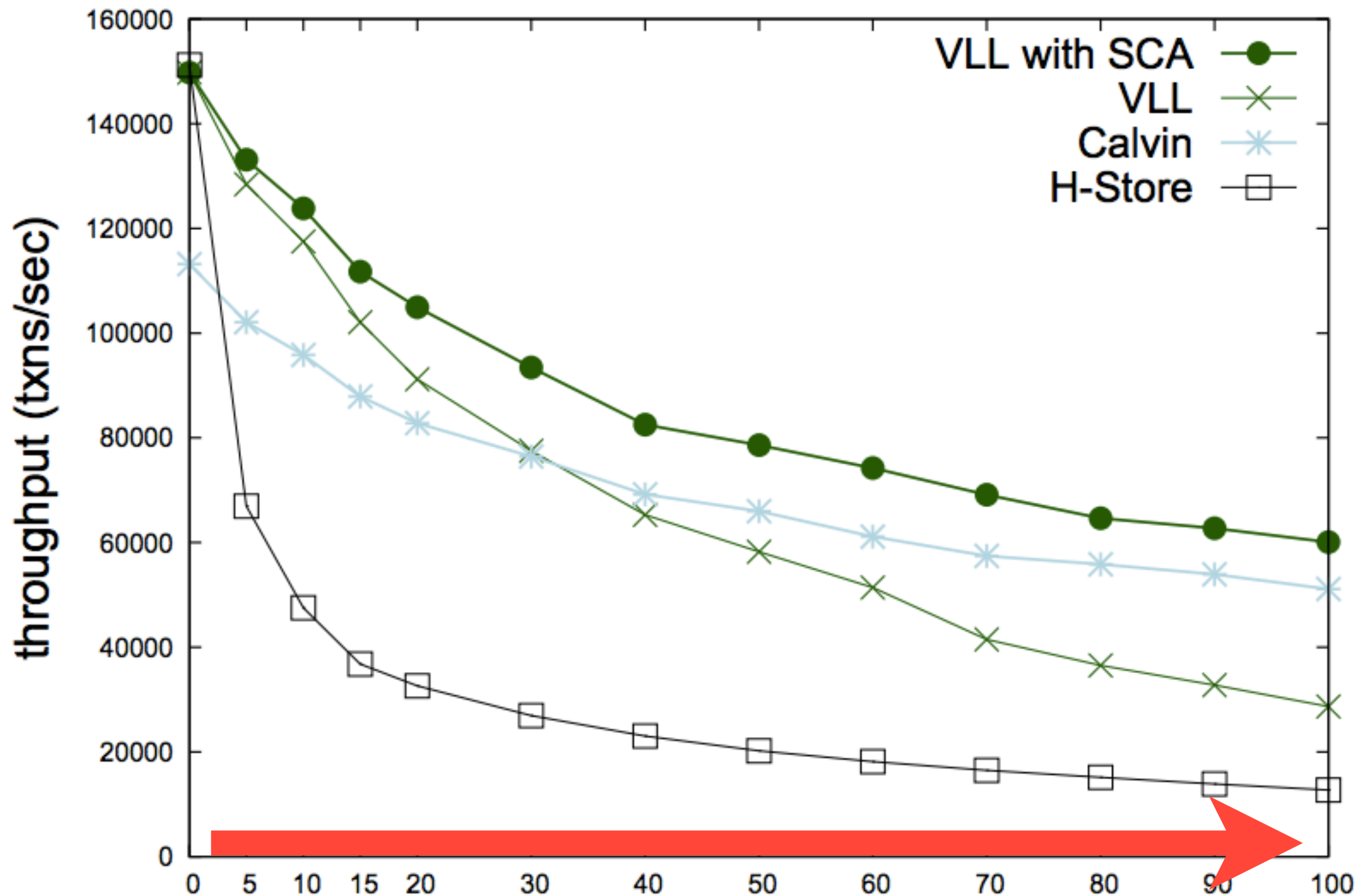
MORE MULTI-PARTITION TXNs

Figure 9: TPC-C throughput.



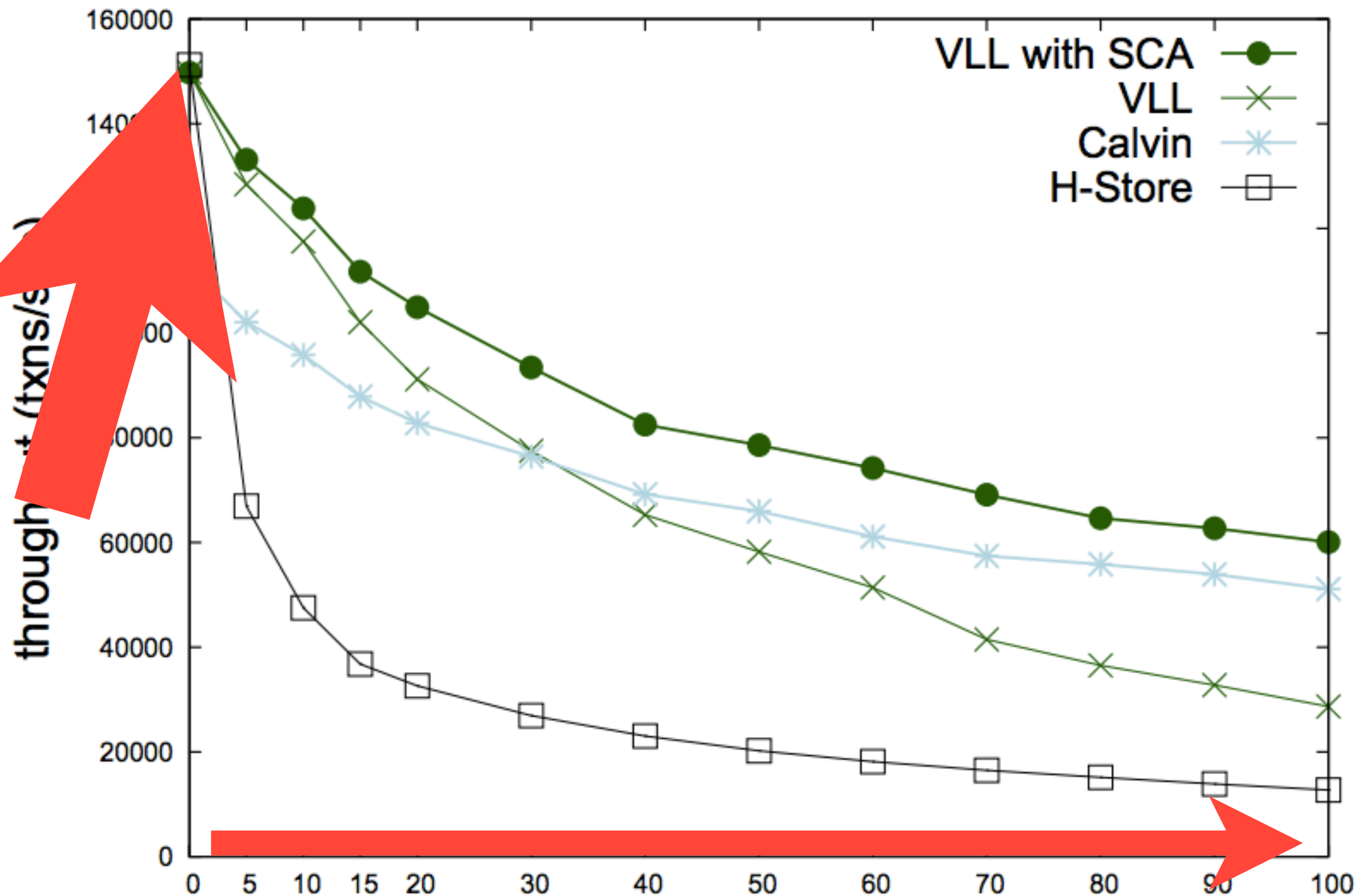
MORE MULTI-PARTITION TXNs

Figure 9: TPC-C throughput.



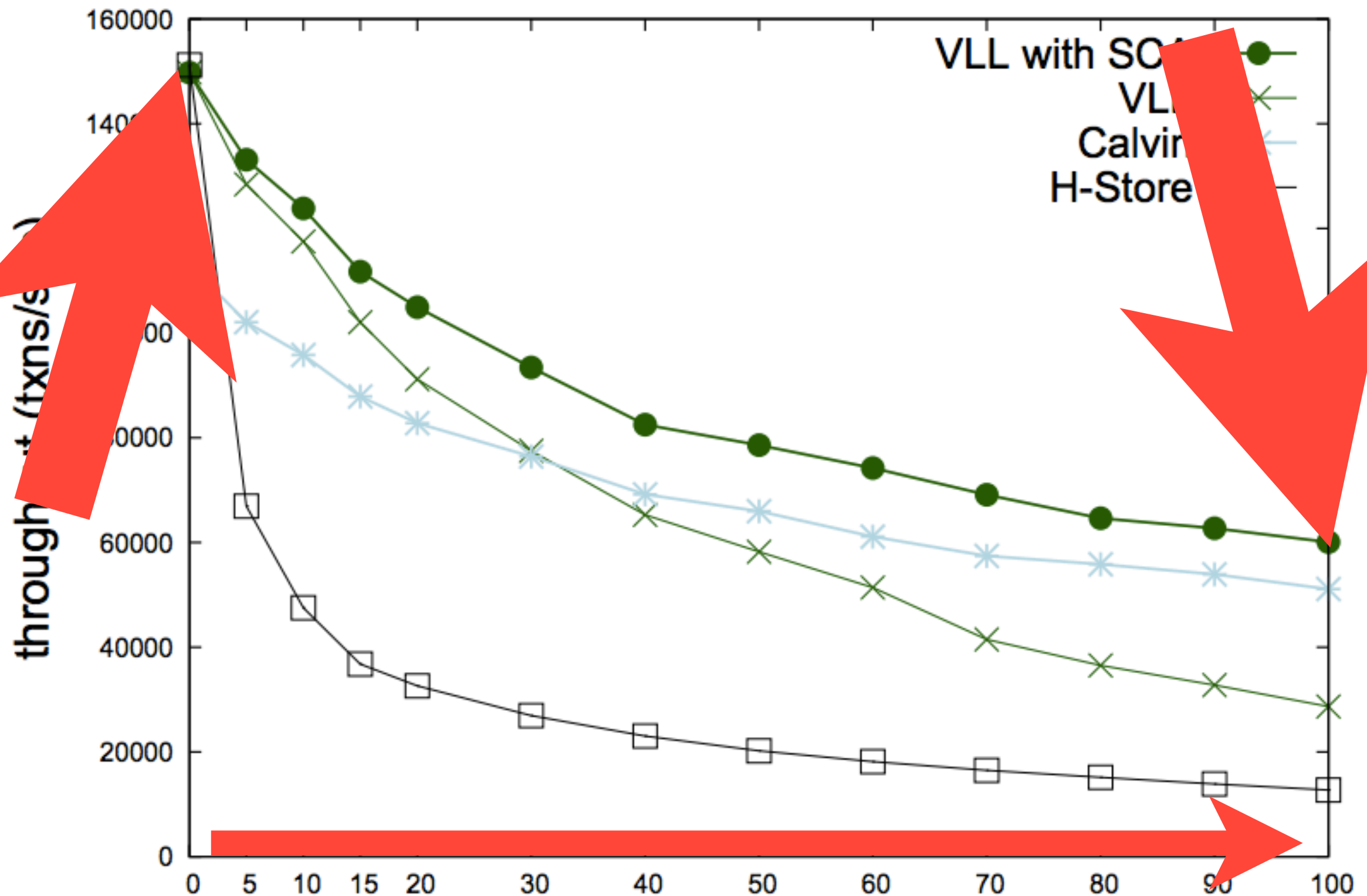
MORE MULTI-PARTITION TXNs

Figure 9: TPC-C throughput.



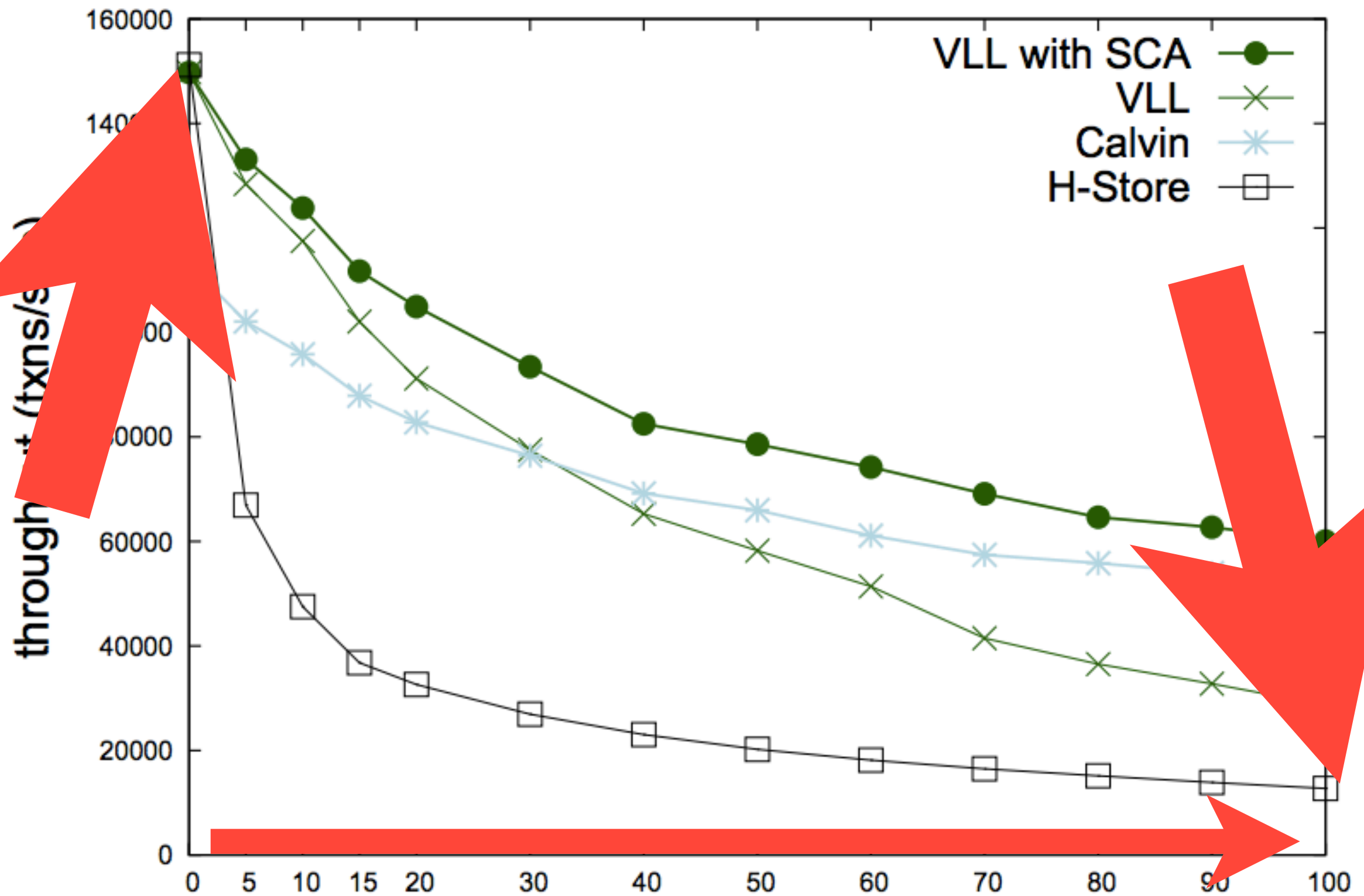
MORE MULTI-PARTITION TXNs

Figure 9: TPC-C throughput.



MORE MULTI-PARTITION TXNs

Figure 9: TPC-C throughput.



MORE MULTI-PARTITION TXNs

Figure 9: TPC-C throughput.

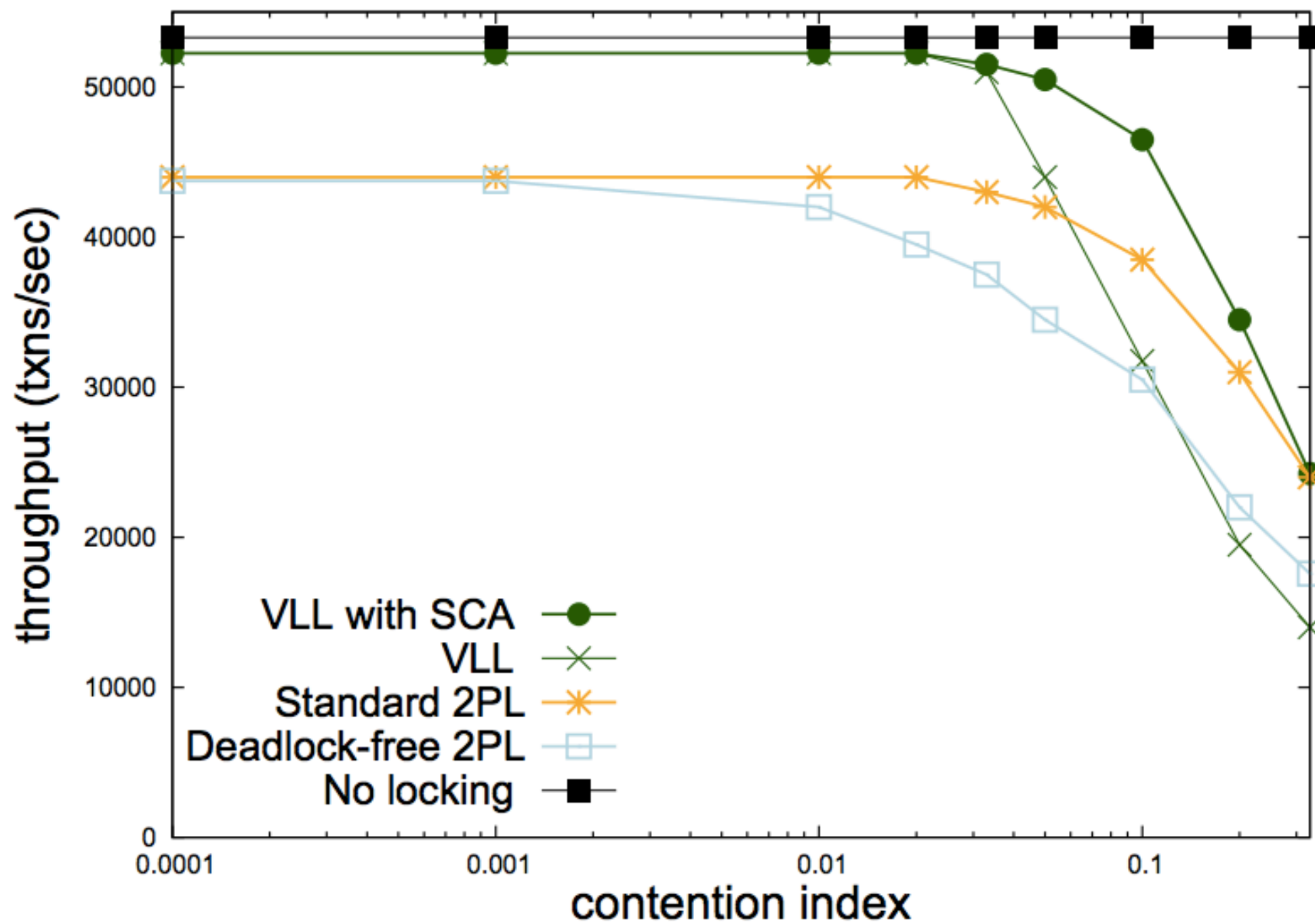


Figure 4: Transactional throughput vs. contention under a deadlock-free workload.

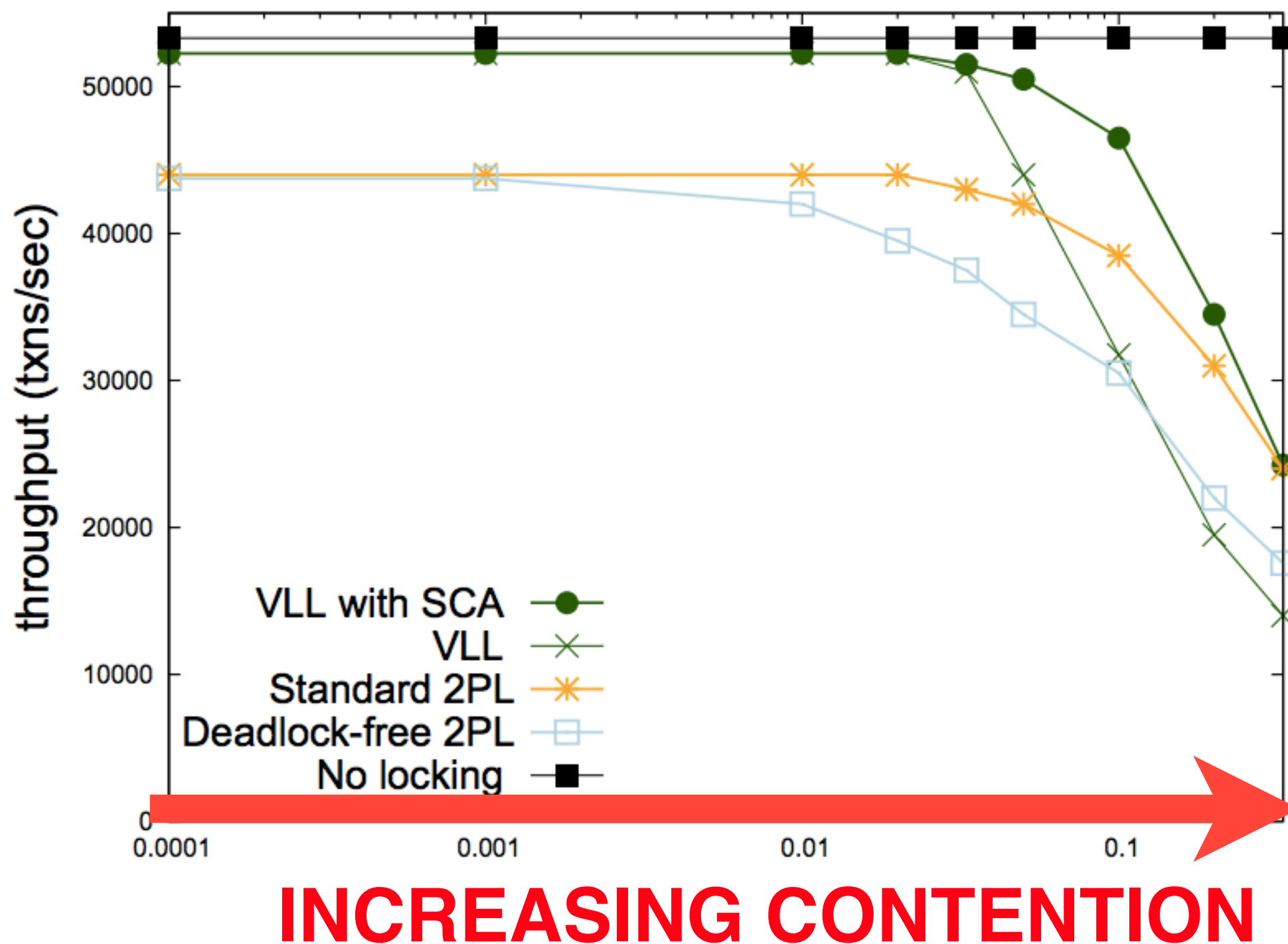


Figure 4: Transactional throughput vs. contention under a deadlock-free workload.

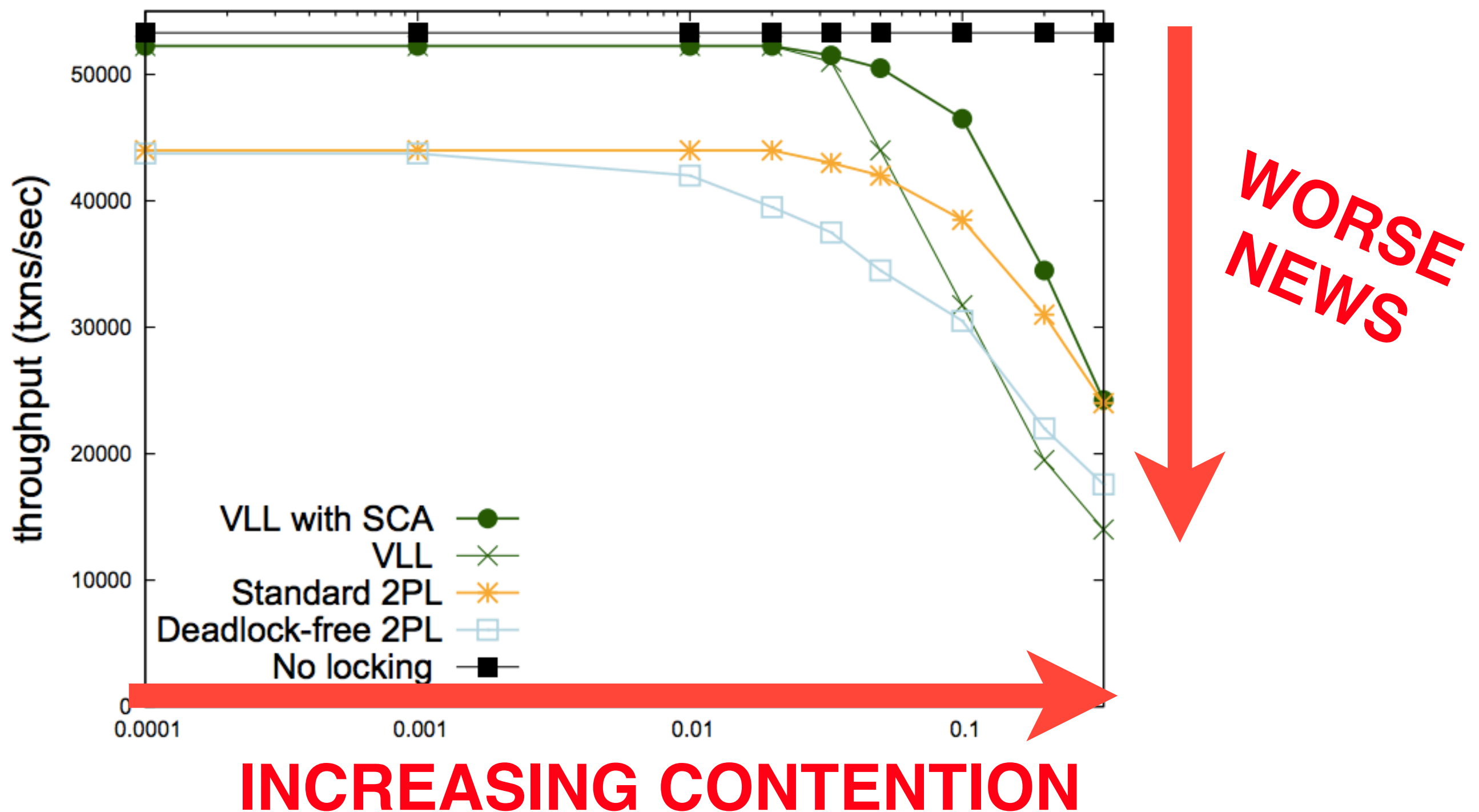


Figure 4: Transactional throughput vs. contention under a deadlock-free workload.

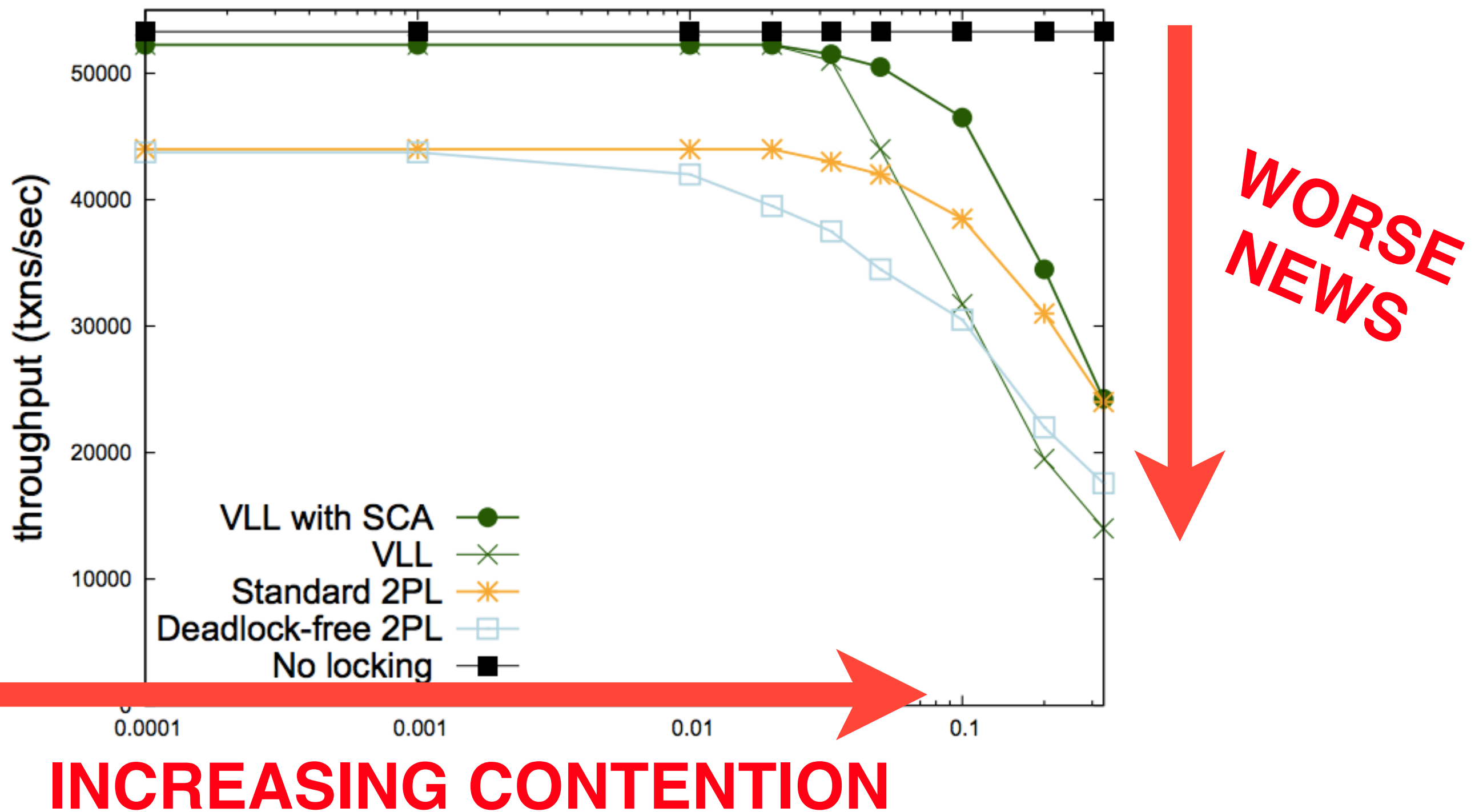
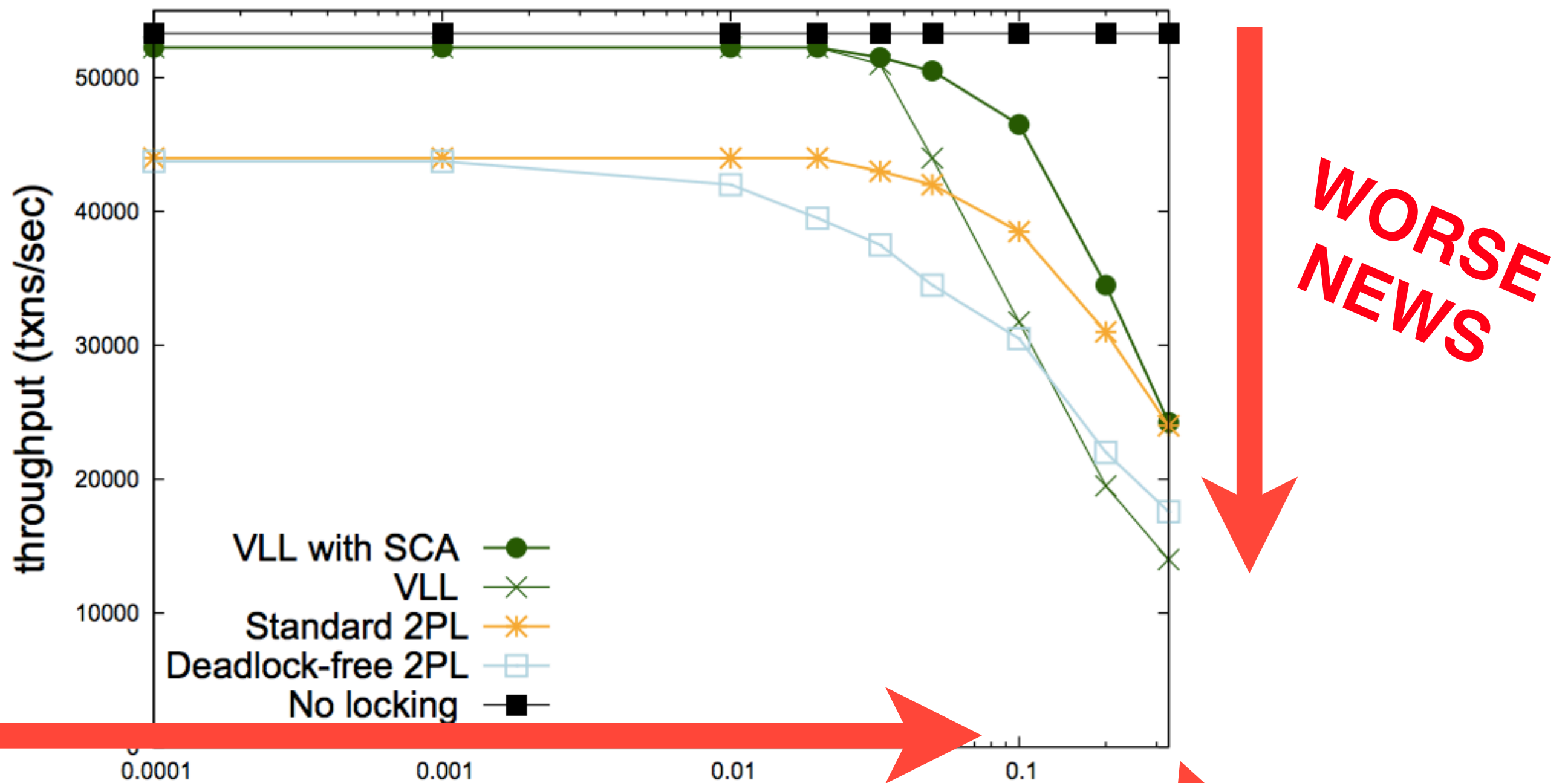


Figure 4: Transactional throughput vs. contention under a deadlock-free workload.



INCREASING CONTENTION

Figure 4: Transactional throughput vs. contention under a deadlock-free workload.

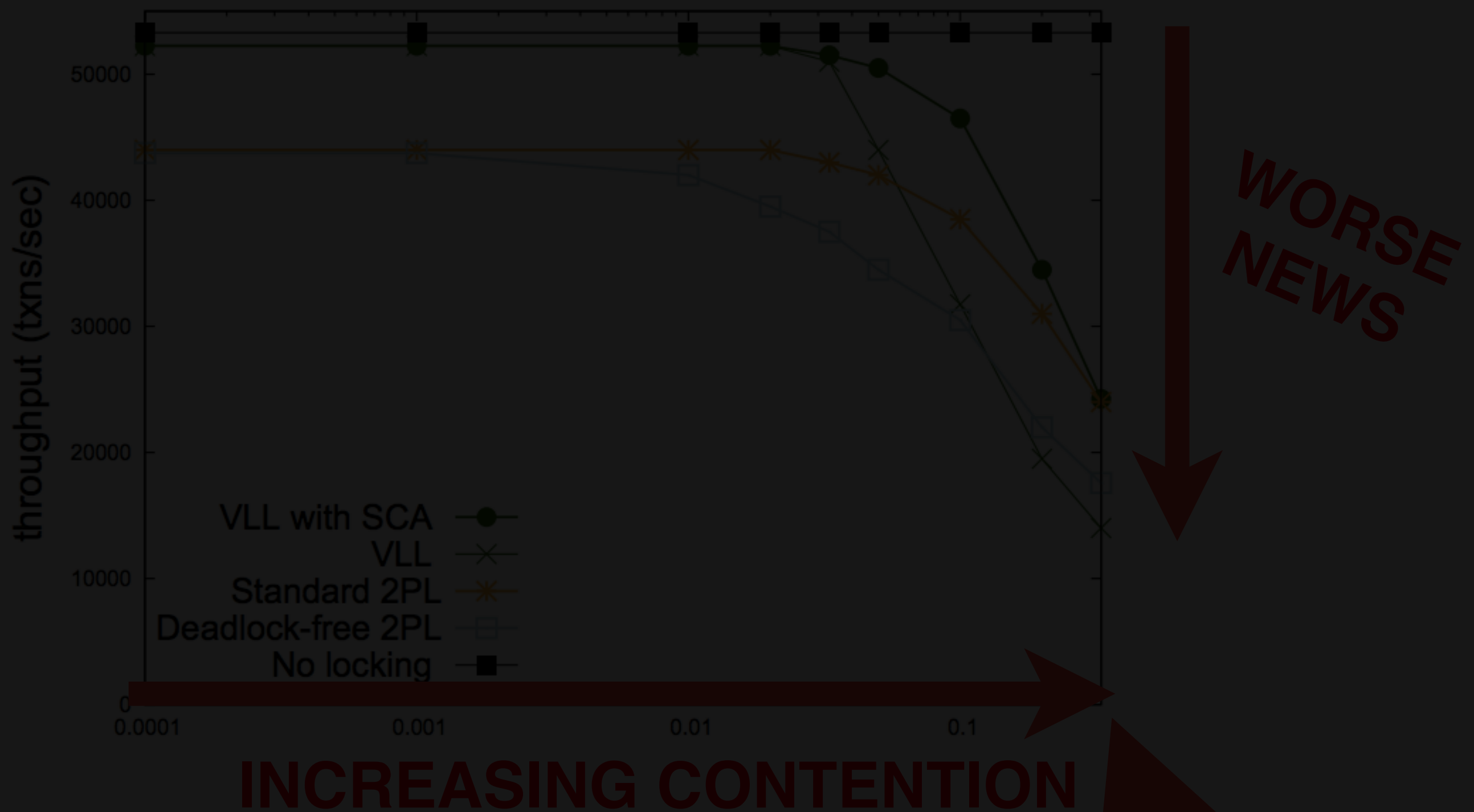


Figure 4: Transactional throughput vs. contention under a deadlock-free workload.

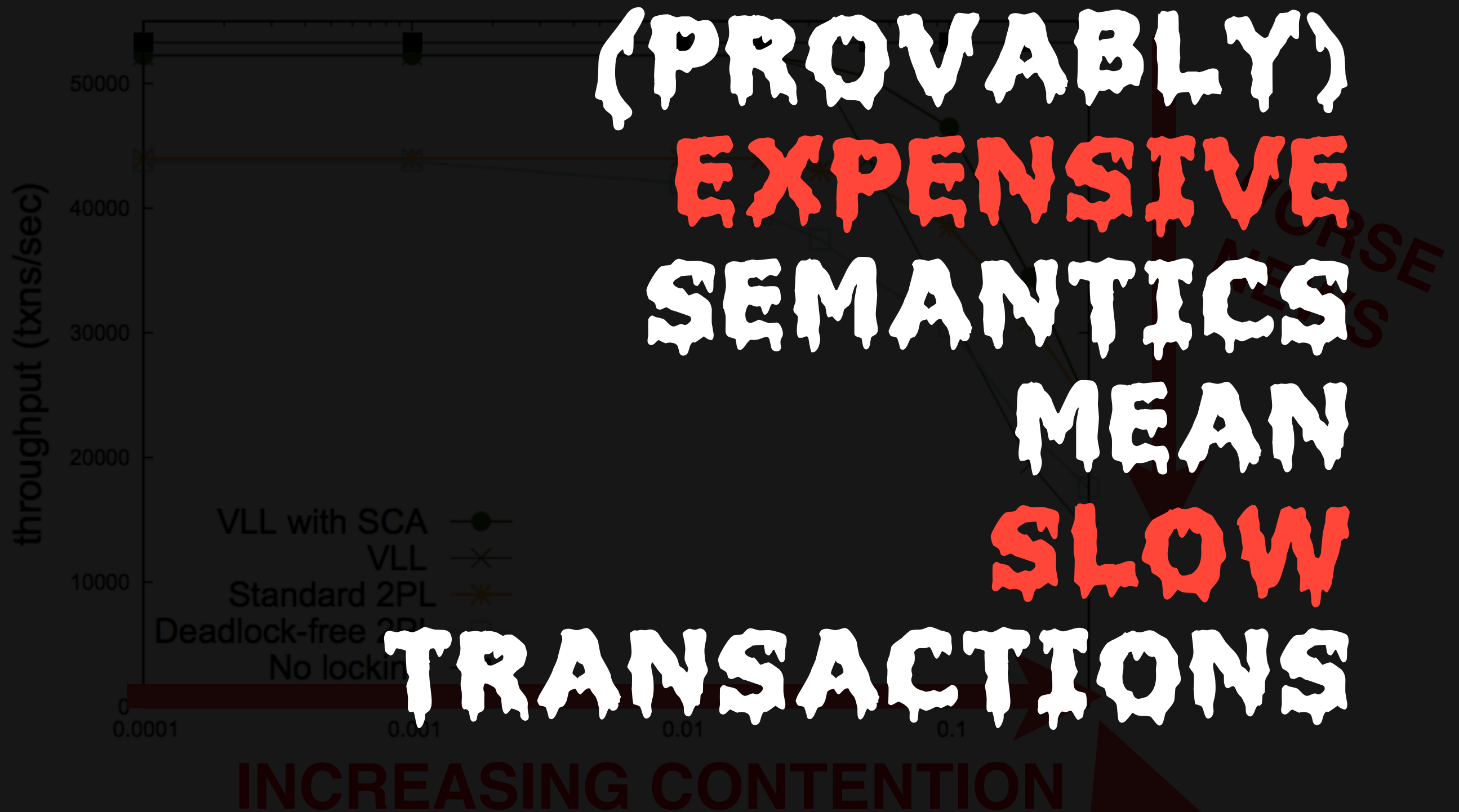
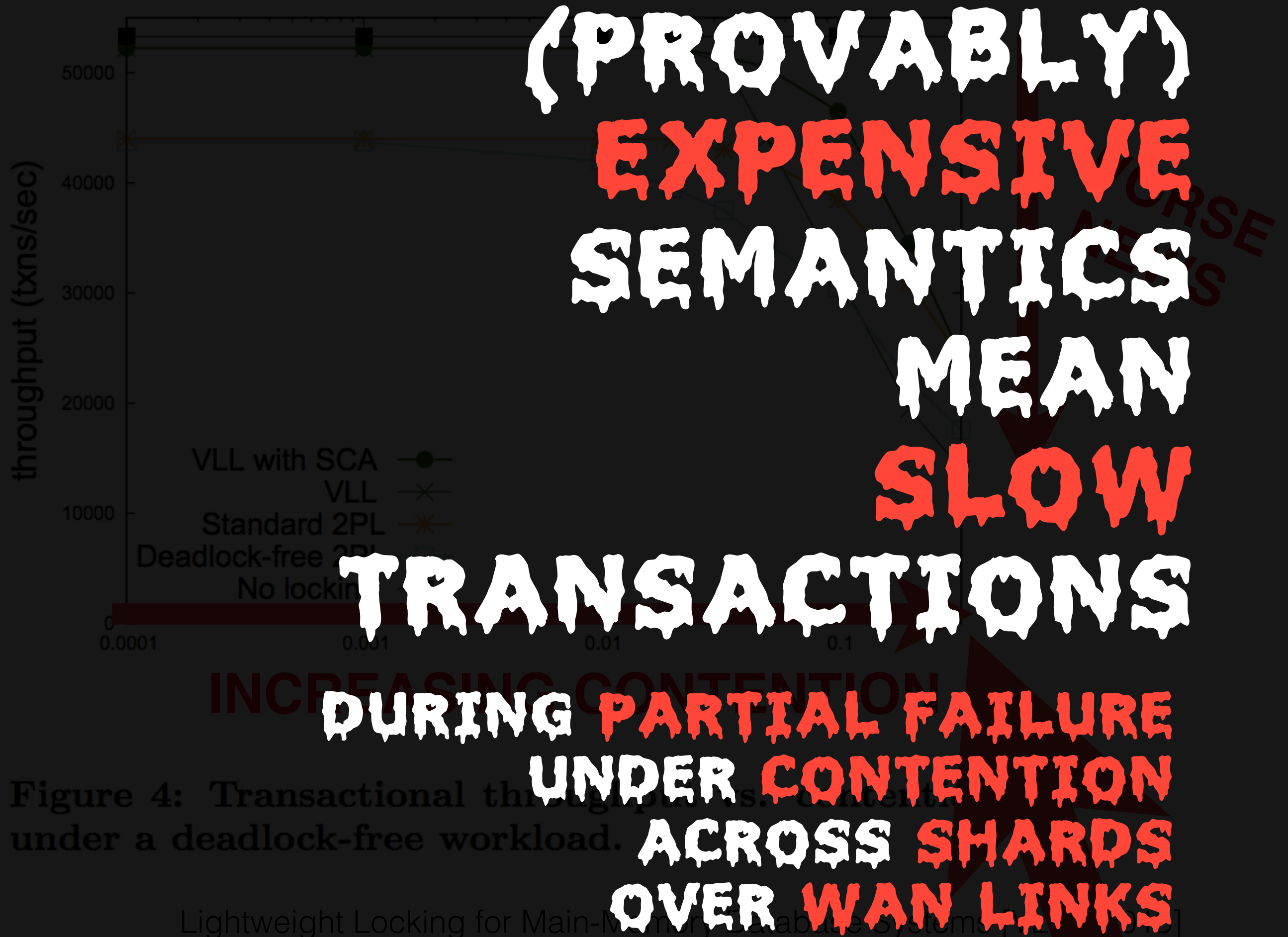


Figure 4: Transactional throughput vs. contention under a deadlock-free workload.





ACID transaction

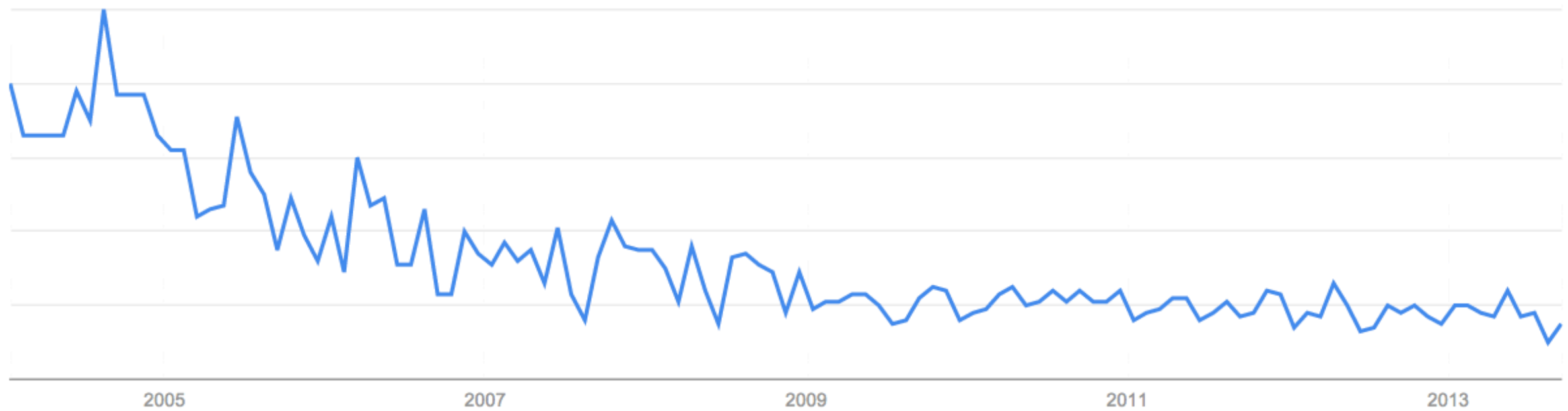
Search term

+ Add term

Interest over time ?

☐ News headlines ?

☐ Forecast ?





ACID transaction

Search term

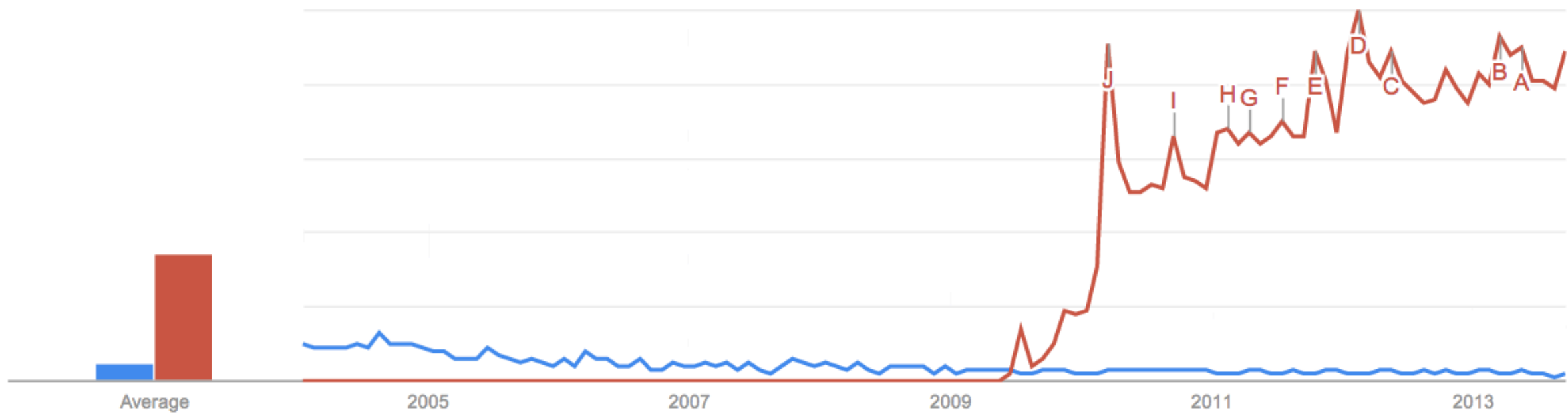
NoSQL

Search term

+ Add term

Interest over time ?

☒ News headlines ☐ Forecast ?





ACID transaction
Search term

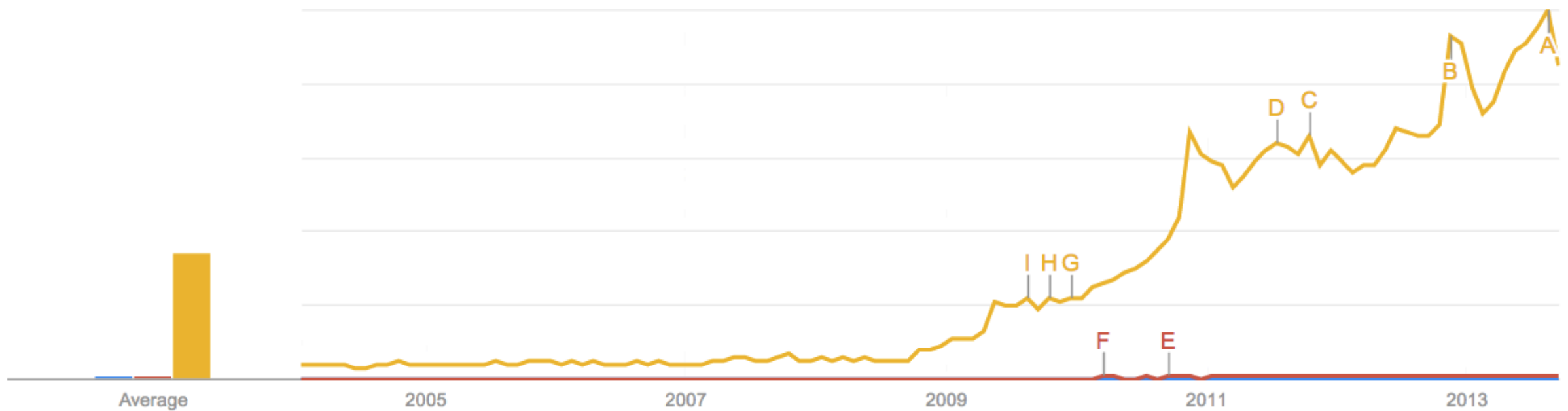
NoSQL
Search term

zombies
Search term

+ Add term

Interest over time ?

☒ News headlines ☐ Forecast ?











web scale database|

web scale database

modern web-scale databases

web scale **graph** database

mongodb is a web scale database





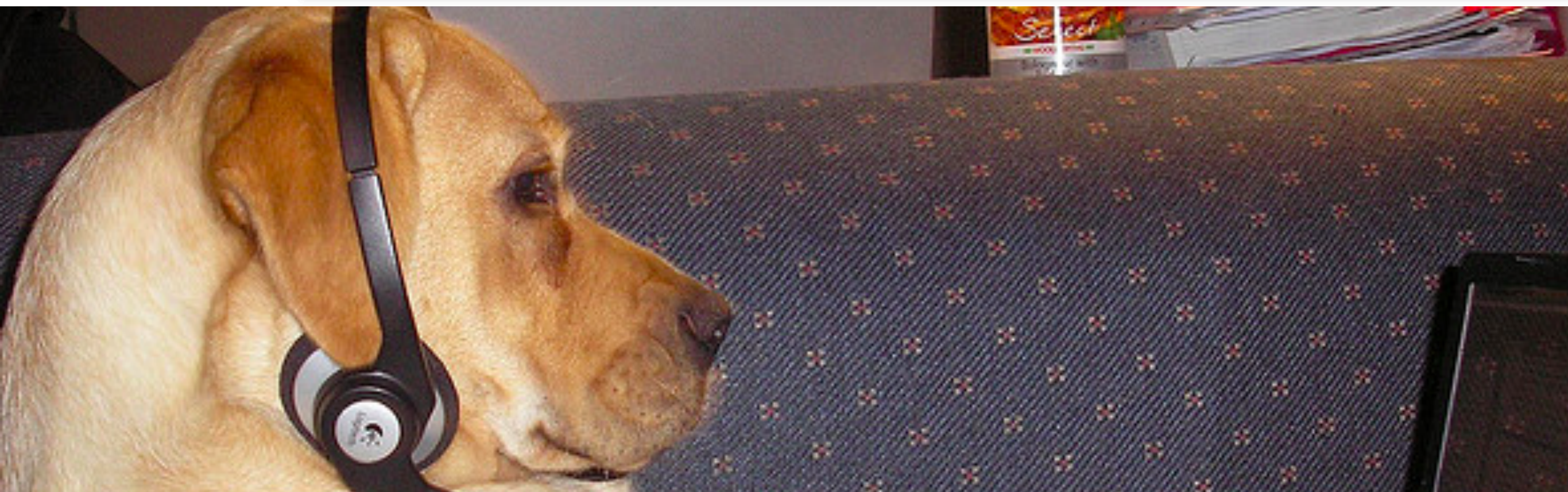
web scale database|

web scale database

modern web-scale databases

web scale **graph** database

mongodb is a web scale database



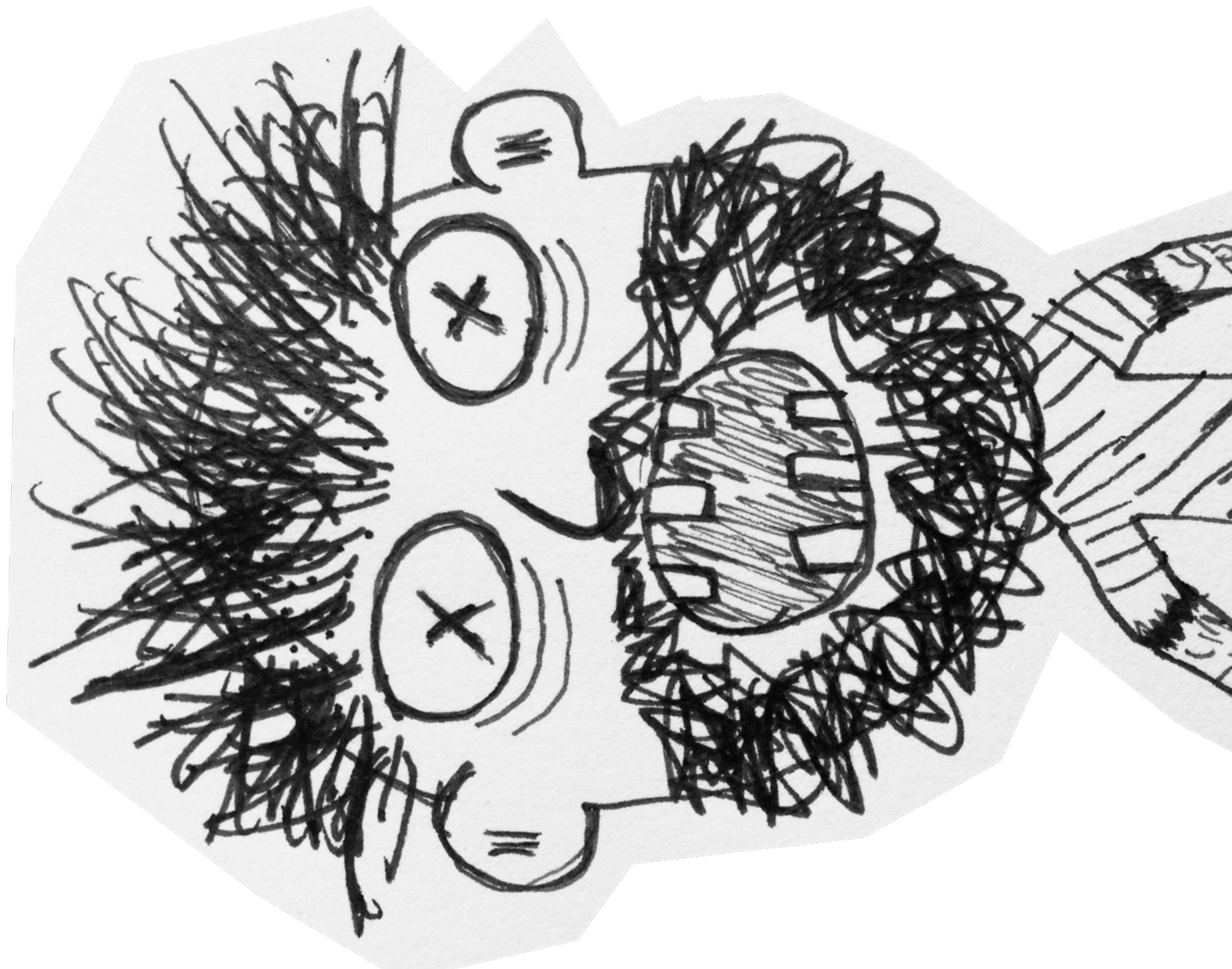
sharding is

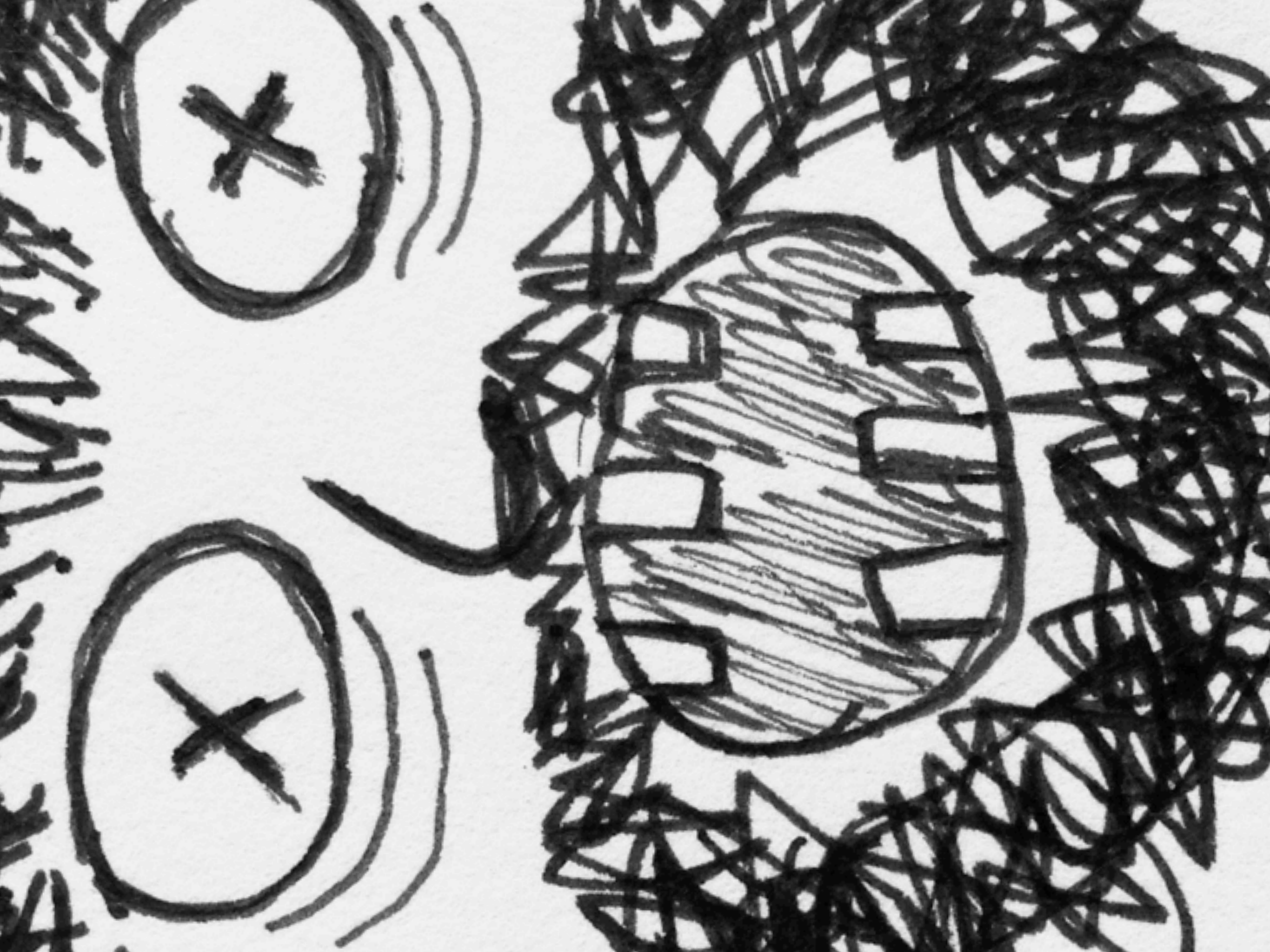
sharding is **a technique** for

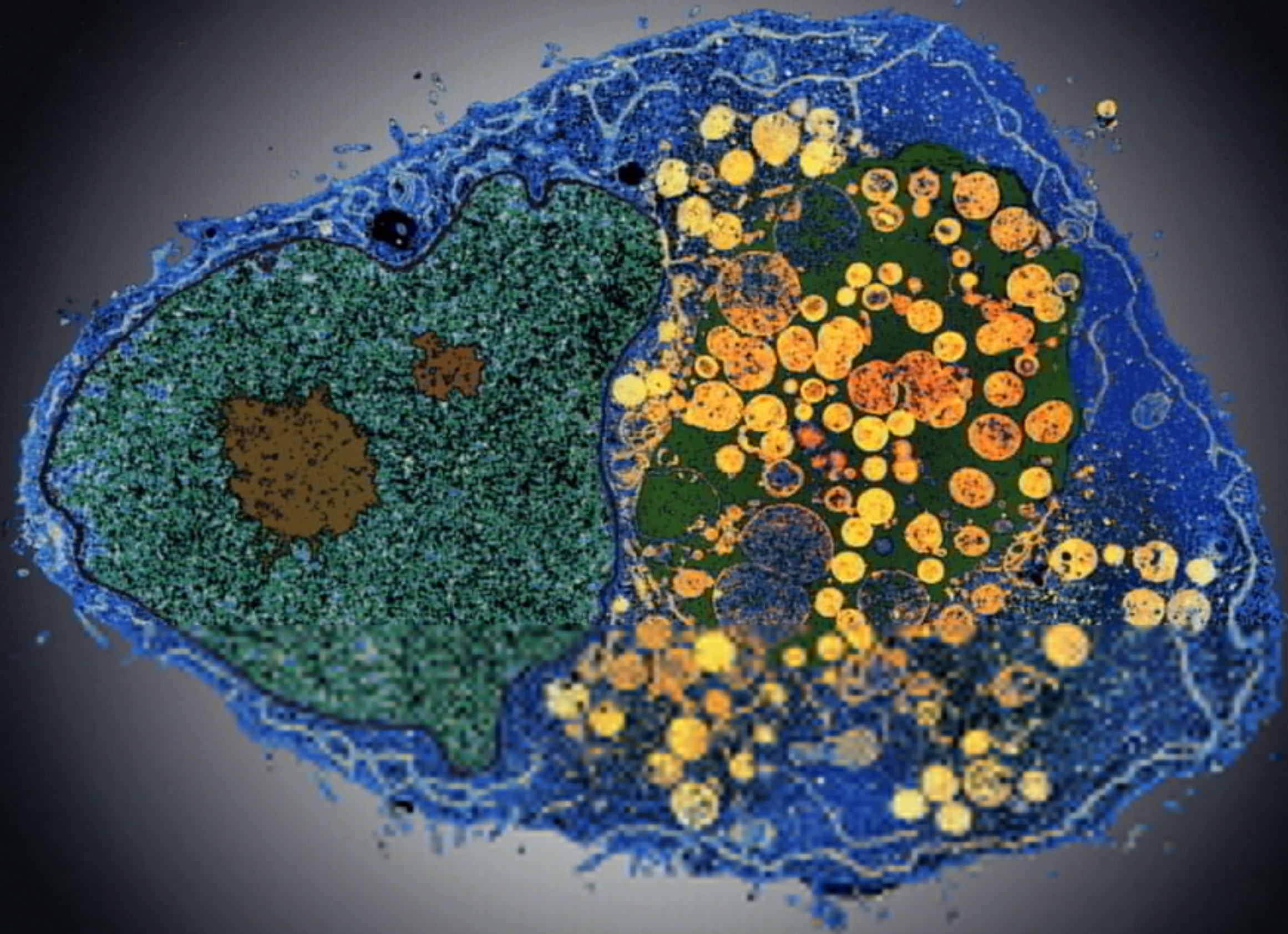
sharding is **the secret sauce**

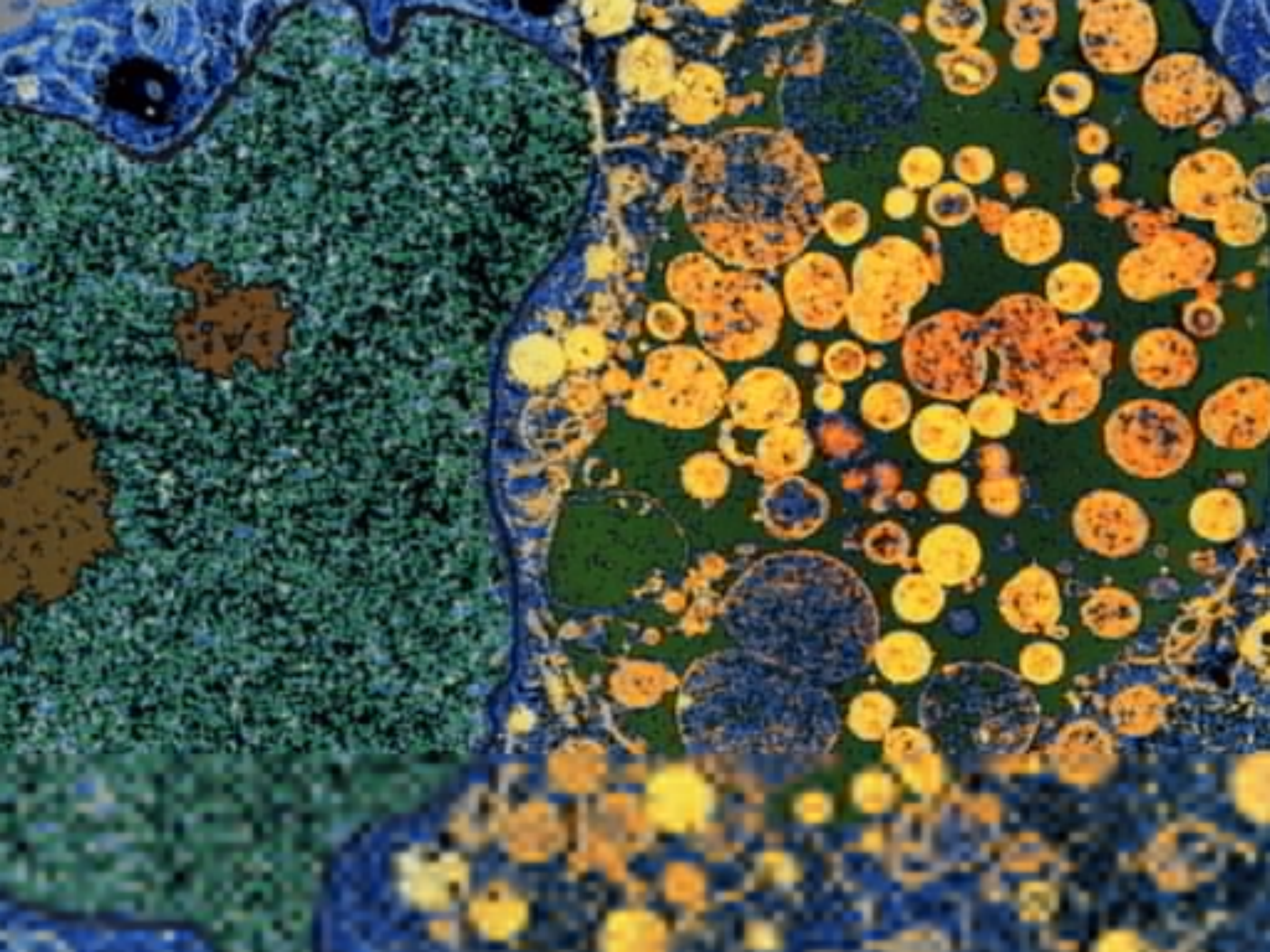
sharding is **the secret ingredient**

sharing is bad



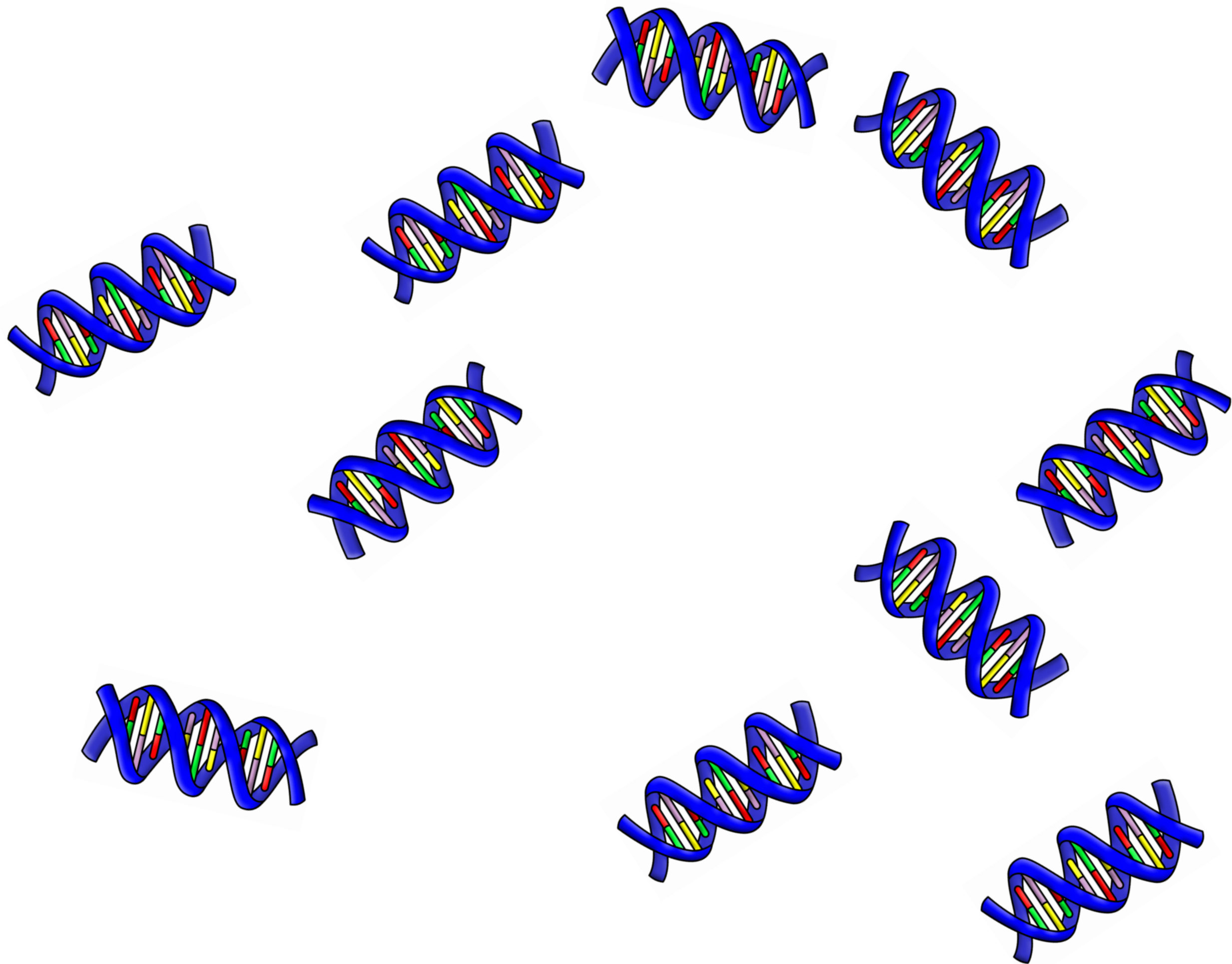


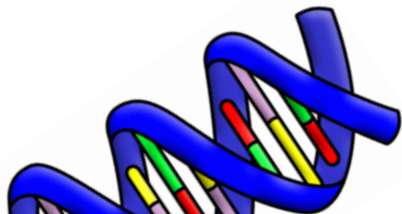
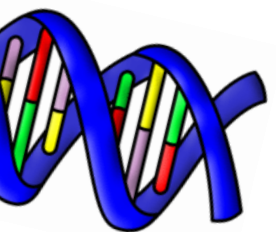
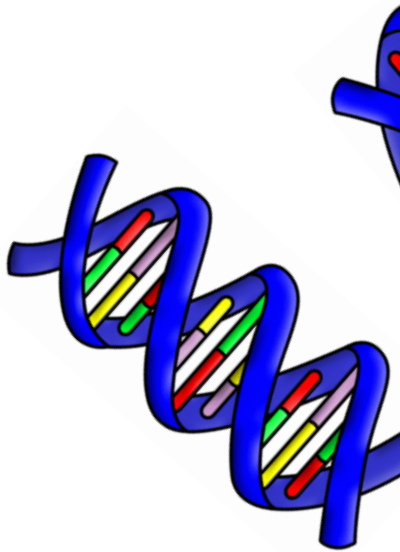
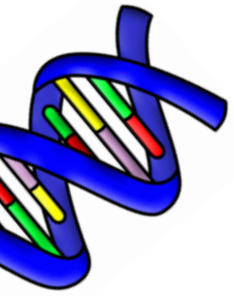
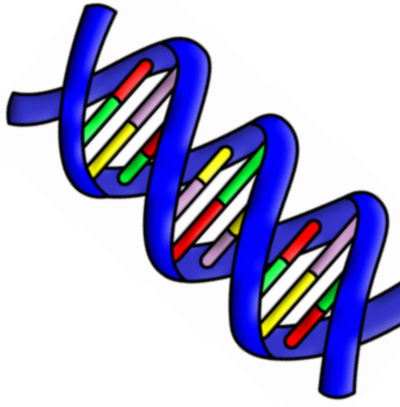
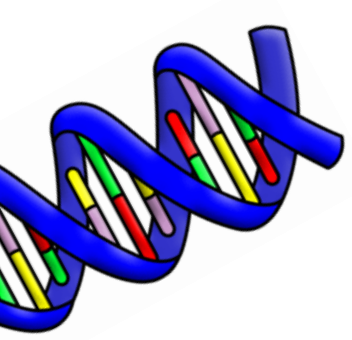


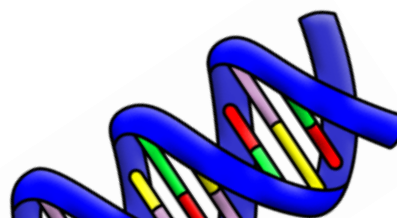
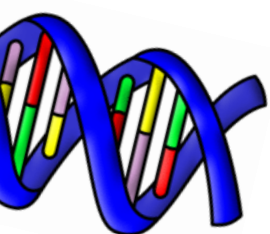
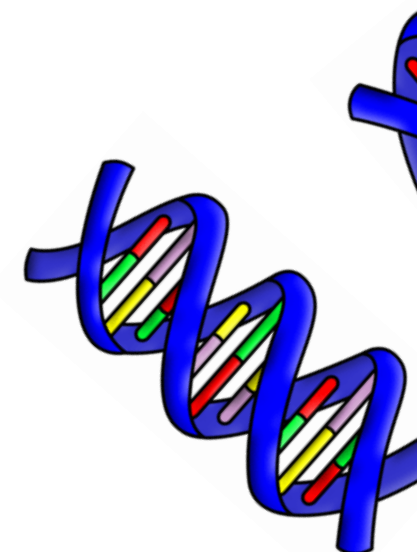
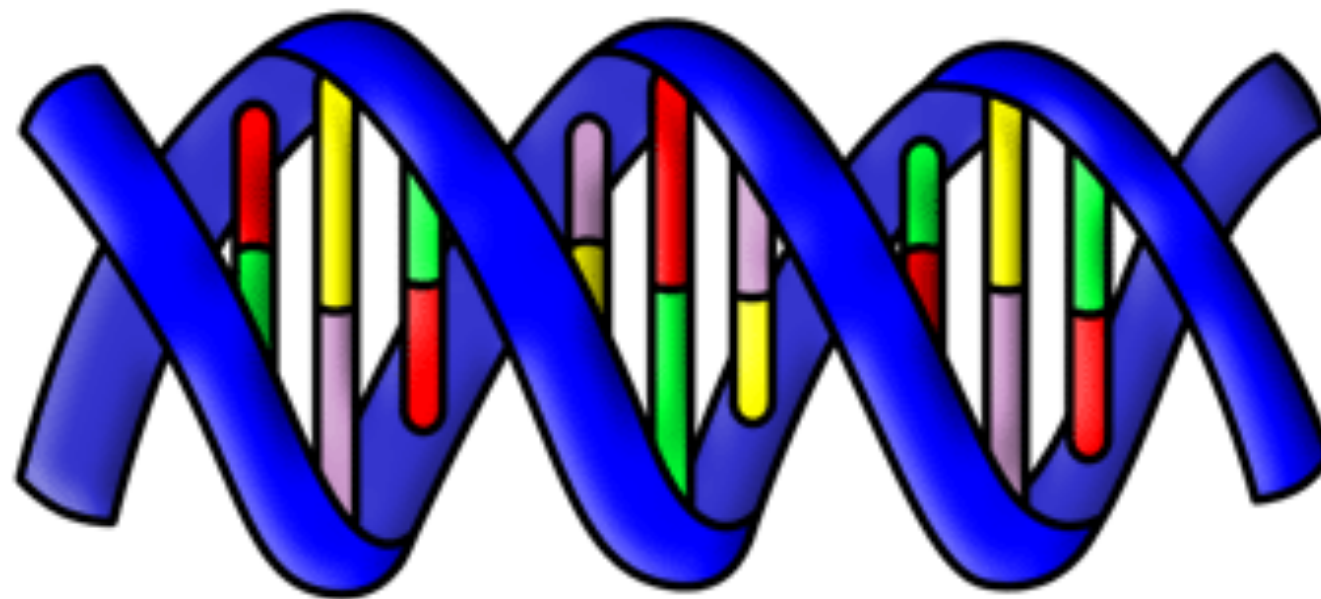
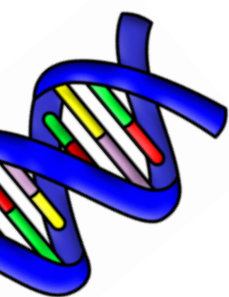
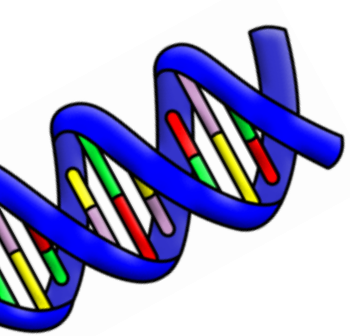


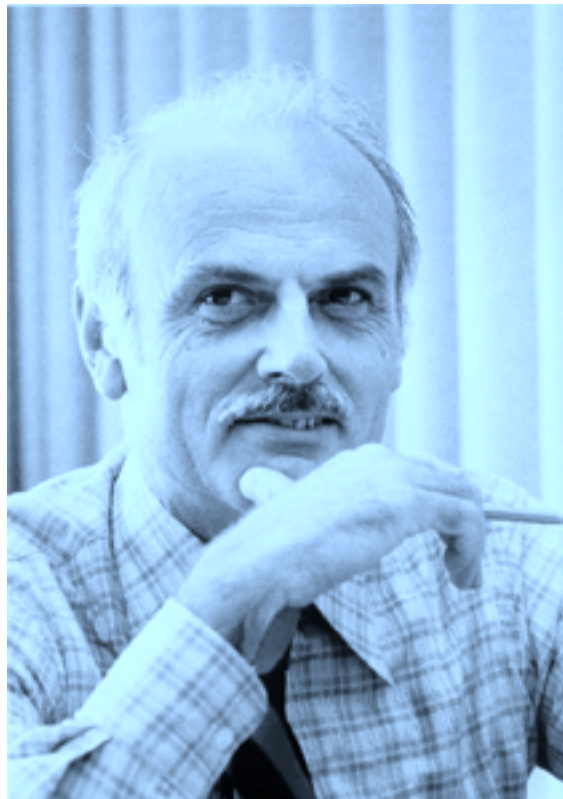
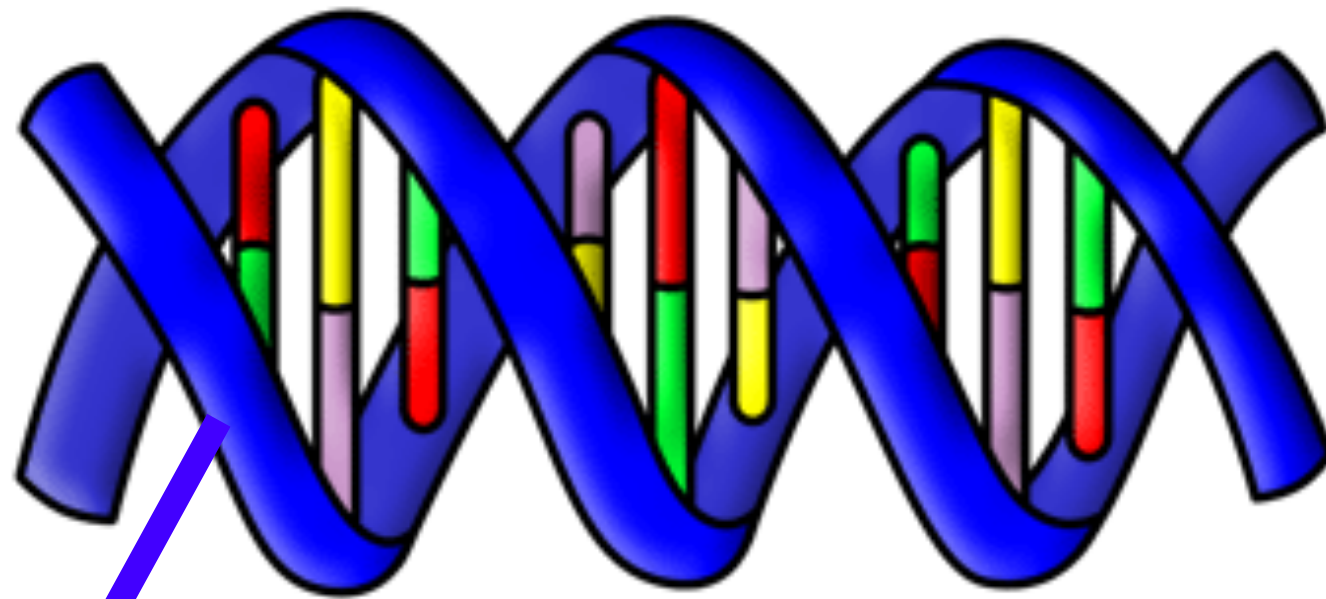


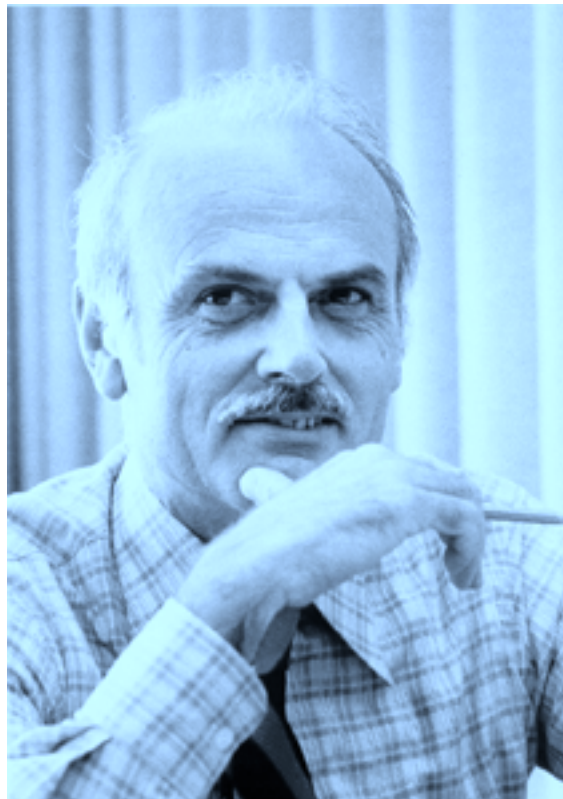
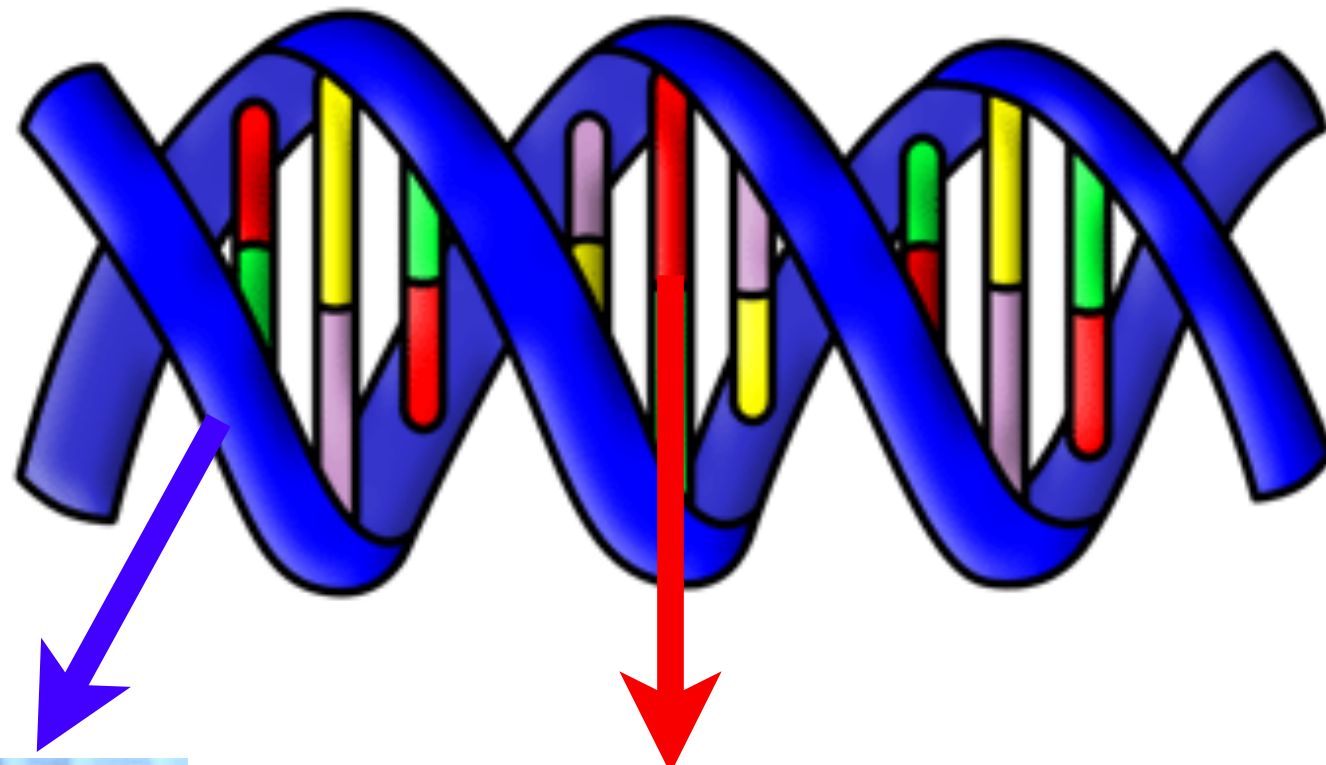


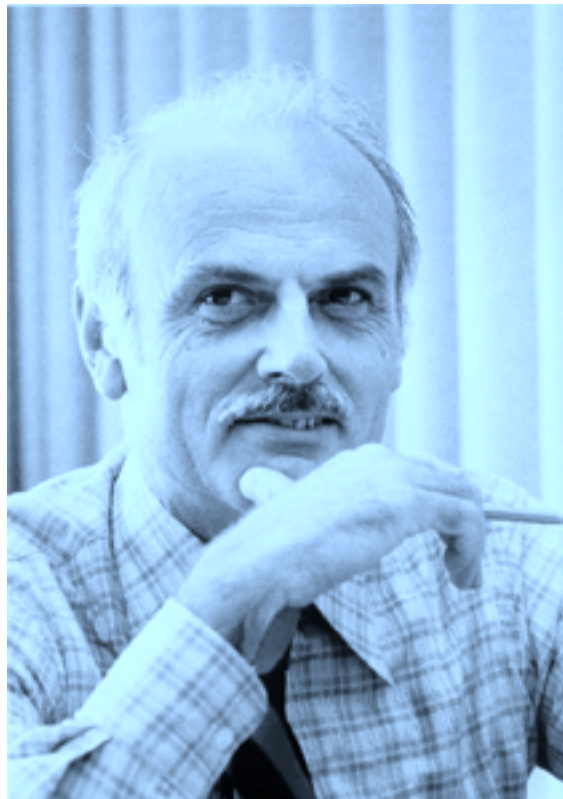
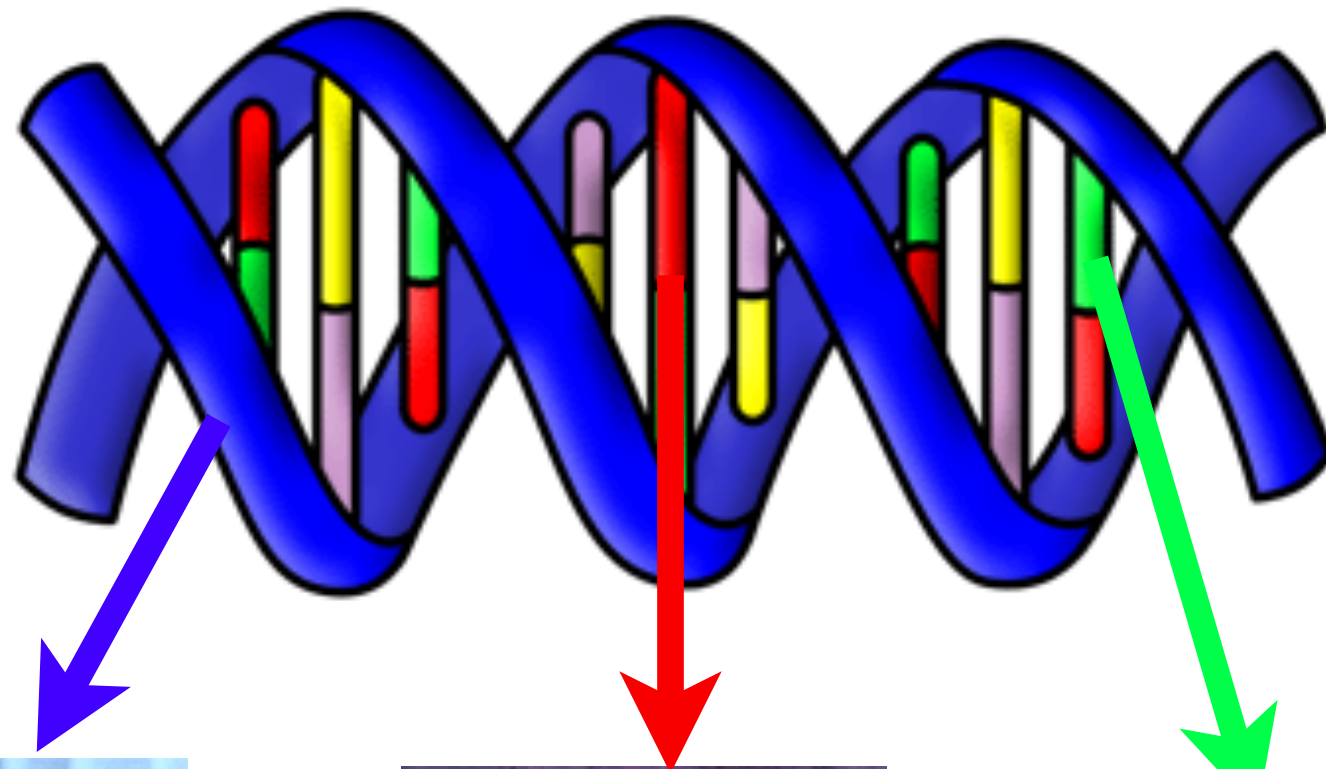












DIAGNOSIS:
coordination overdose

DIAGNOSIS:

coordination overdose

CHALLENGE:

existing **models** failed (conflict serializability)

DURING PARTIAL FAILURE

UNDER CONTENTION

ACROSS SHARDS

OVER WAN LINKS

DIAGNOSIS:

coordination overdose

CHALLENGE:

existing **models** failed (conflict serializability)

DURING PARTIAL FAILURE

UNDER CONTENTION

ACROSS SHARDS

OVER WAN LINKS

scalability problems resulted from **model**,
not from bad implementations

DIAGNOSIS:

coordination overdose

CHALLENGE:

existing **models** failed (conflict serializability)

DURING PARTIAL FAILURE

UNDER CONTENTION

ACROSS SHARDS

OVER WAN LINKS

scalability problems resulted from **model**,
not from bad implementations

so how did we limit coordination but maintain
correctness and programmability?

DIAGNOSIS:
coordination overdose

DIAGNOSIS:

coordination overdose

TREATMENT:

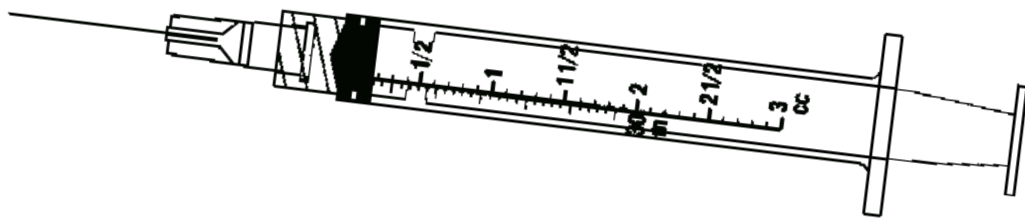
ensure correctness via app-level semantics

DIAGNOSIS:

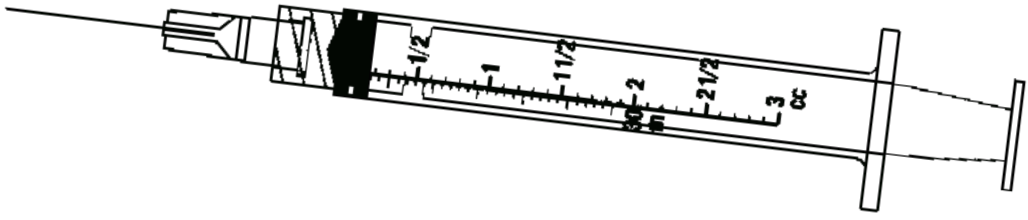
coordination overdose

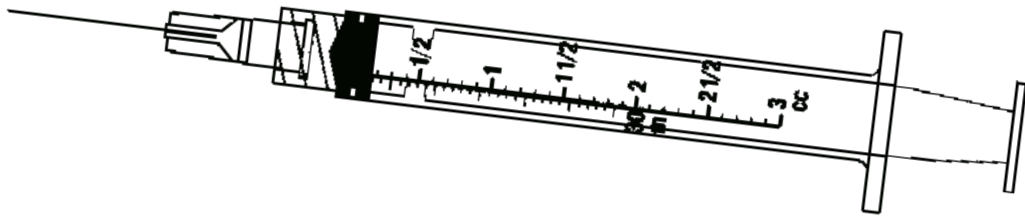
TREATMENT:

ensure correctness via app-level semantics

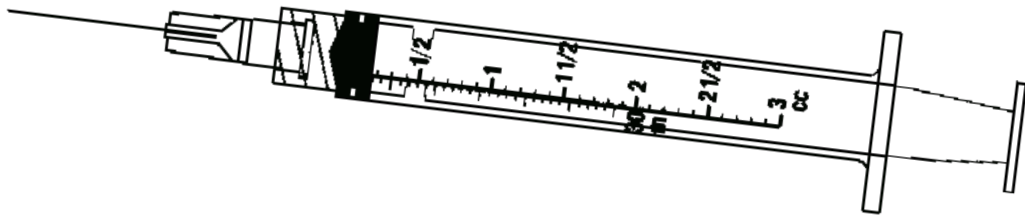


**e.g., use provided constraints
to ensure ACID consistency
with minimal coordination**



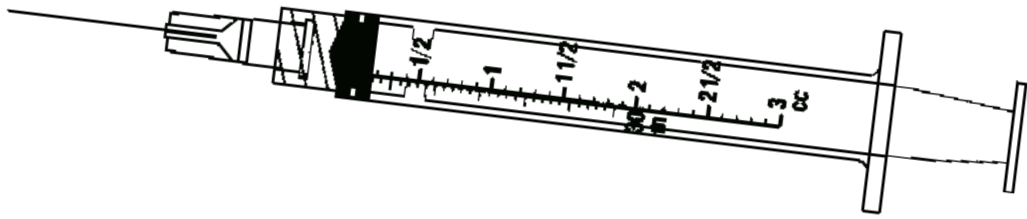


Balanced Concurrency Control



Balanced Concurrency Control

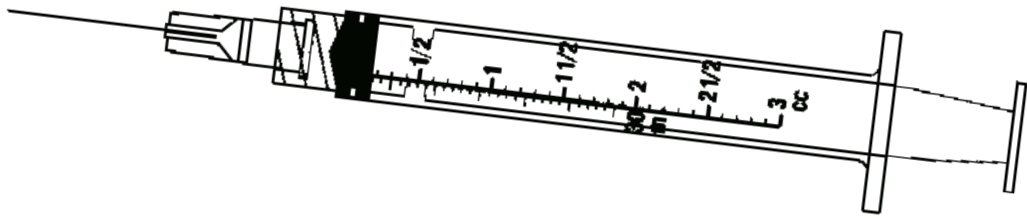
1. Let concurrency flourish.



Balanced Concurrency Control

1. Let concurrency flourish.

Theorem: if transactions commute under invariants, can execute concurrently, without coordination.

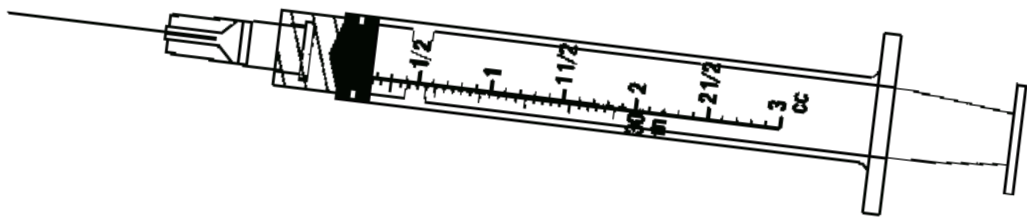


Balanced Concurrency Control

1. Let concurrency flourish.

Theorem: if transactions commute under invariants, can execute concurrently, without coordination.

2. Minimize distribution (space and time) of conflicting transactions.



Balanced Concurrency Control

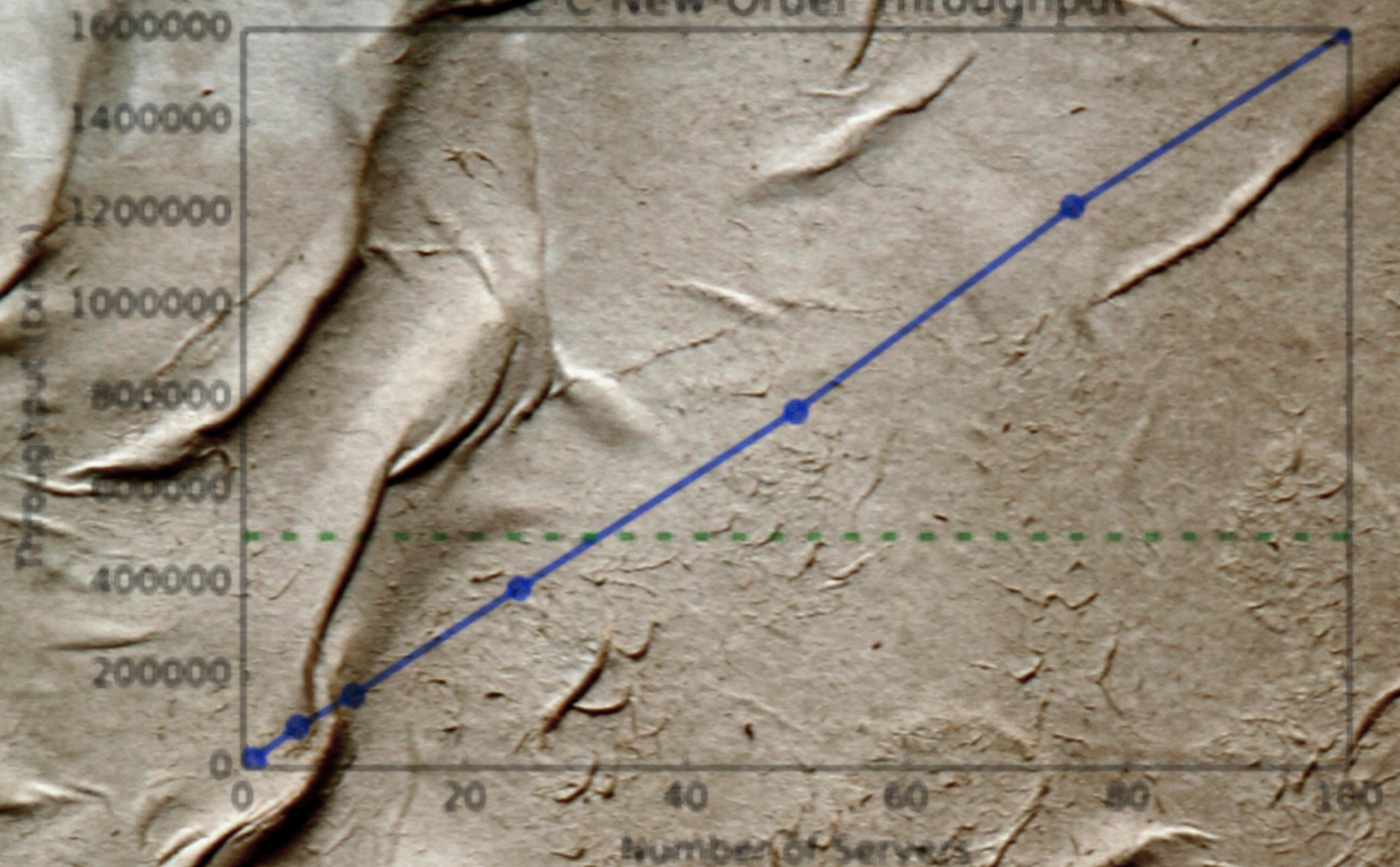
1. Let concurrency flourish.

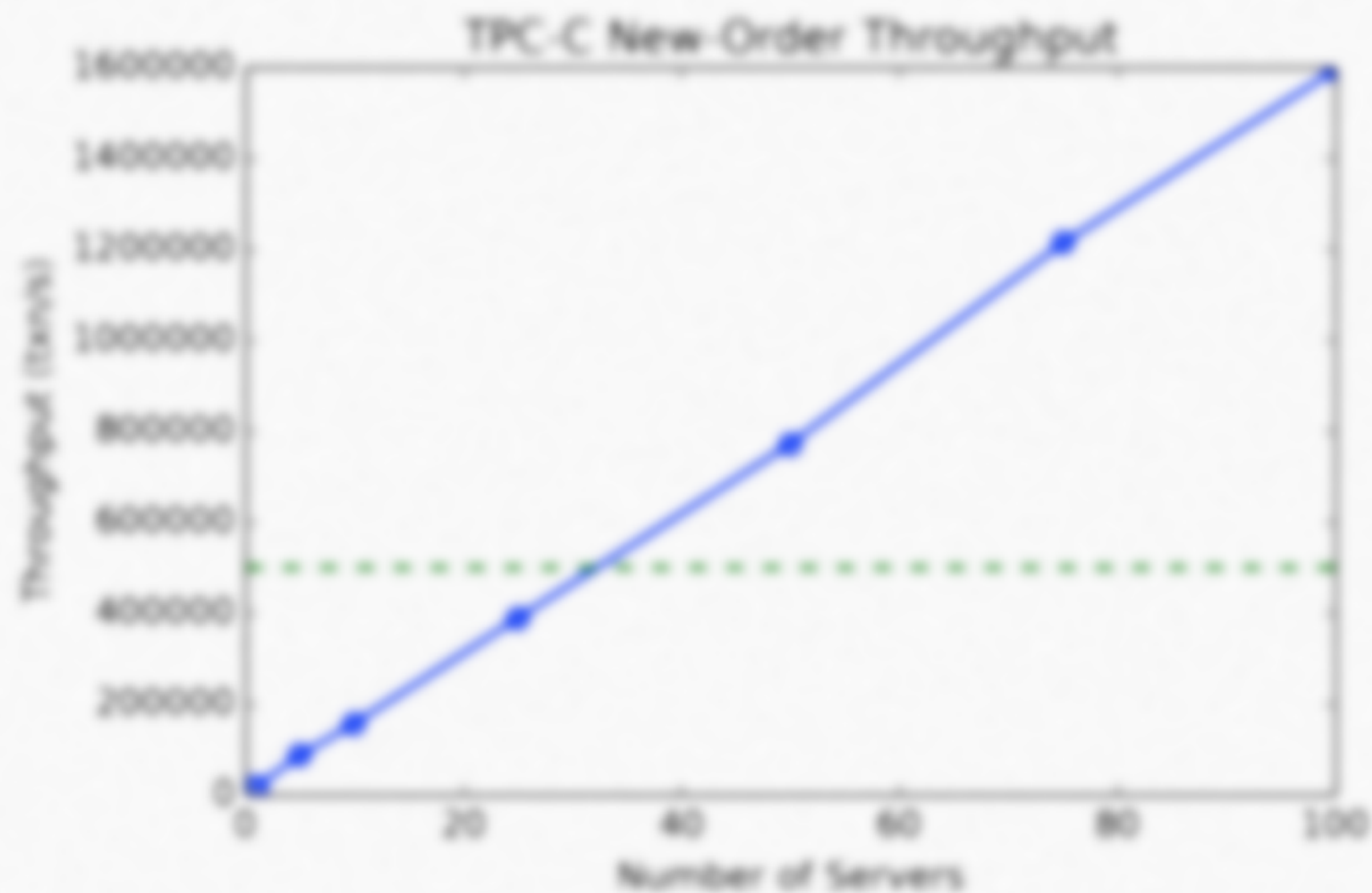
Theorem: if transactions commute under invariants, can execute concurrently, without coordination.

2. Minimize distribution (space and time) of conflicting transactions.

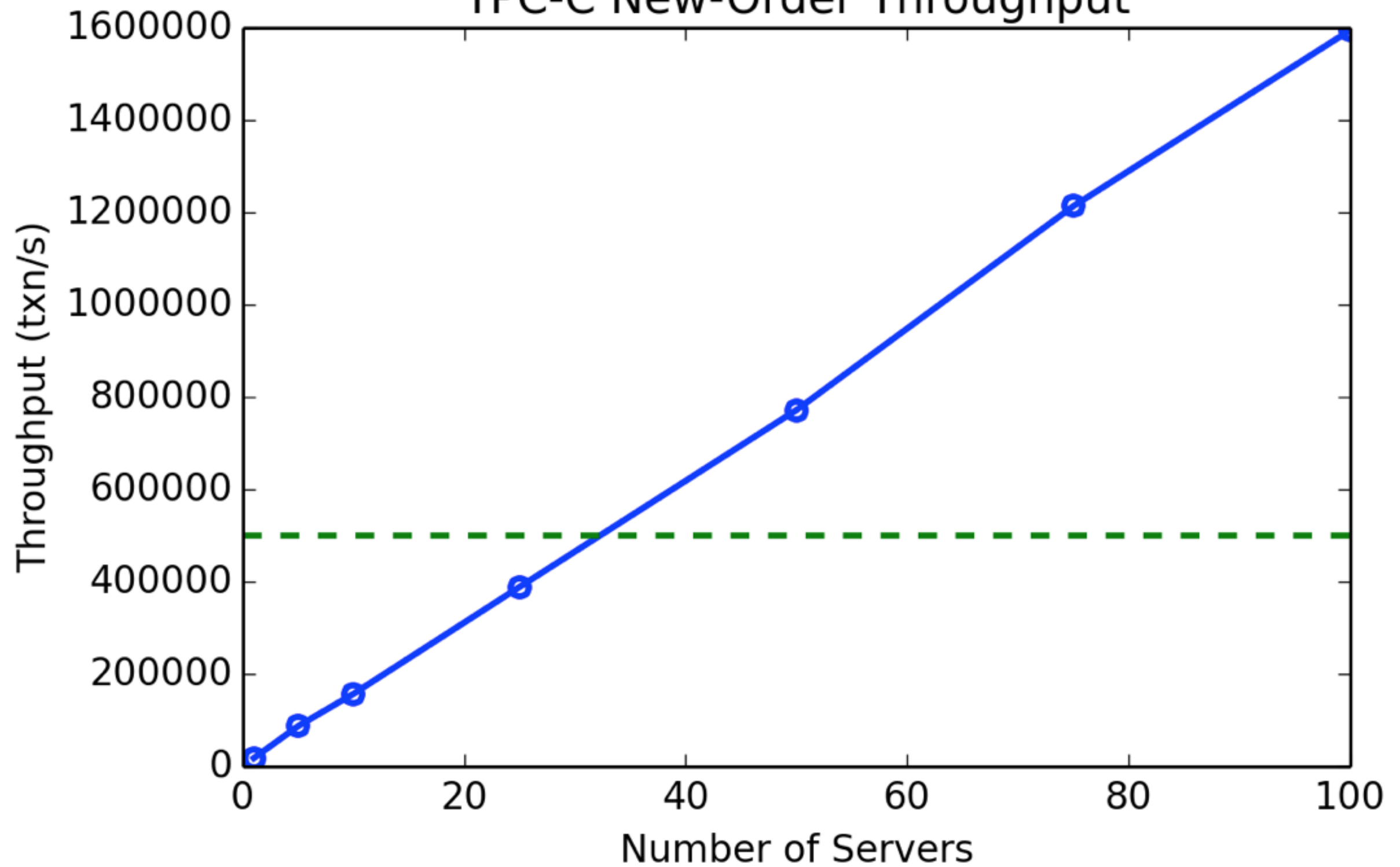
Perform query analysis and rewriting to limit coordination between processes.

TPC-C New-Order Throughput

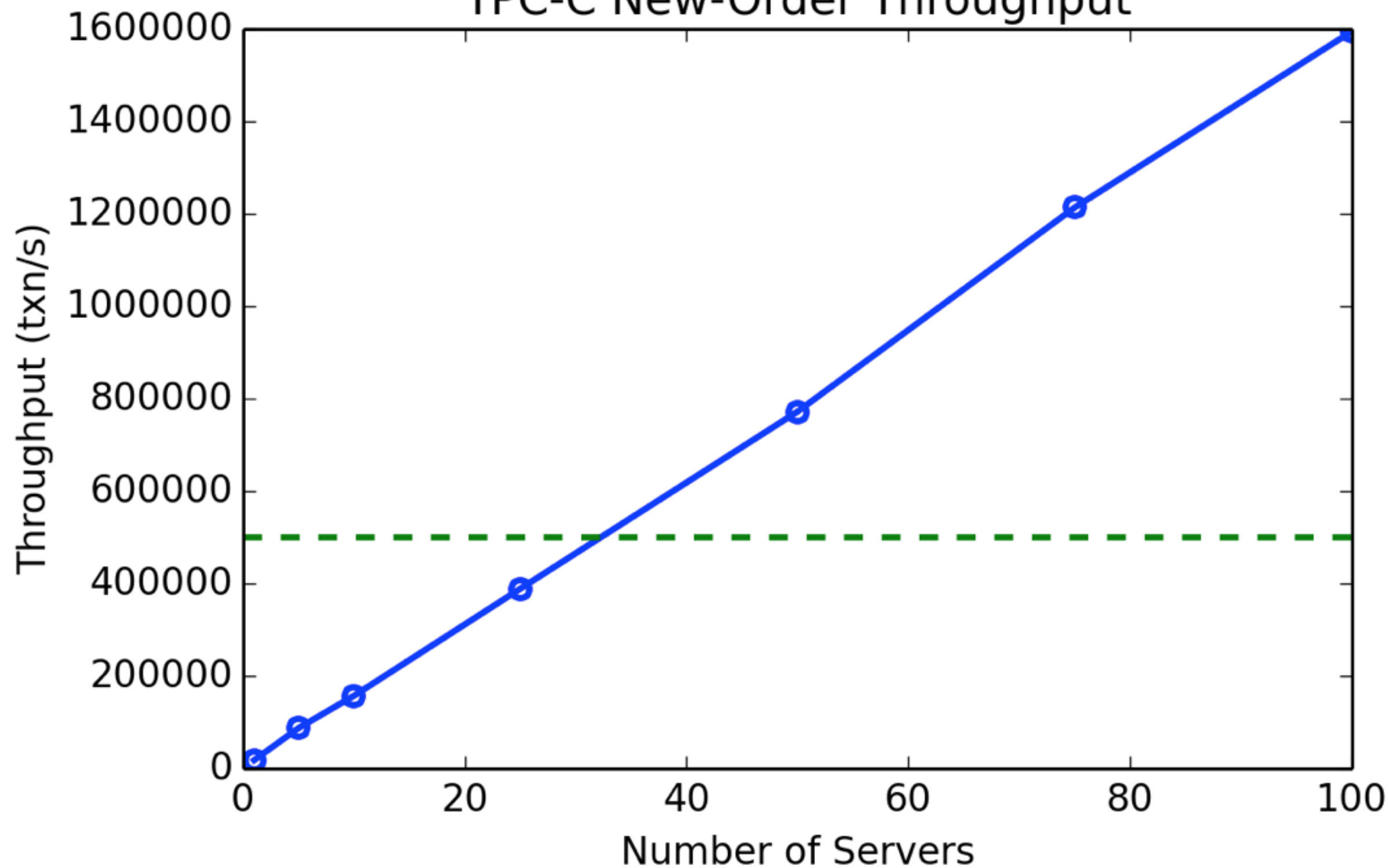




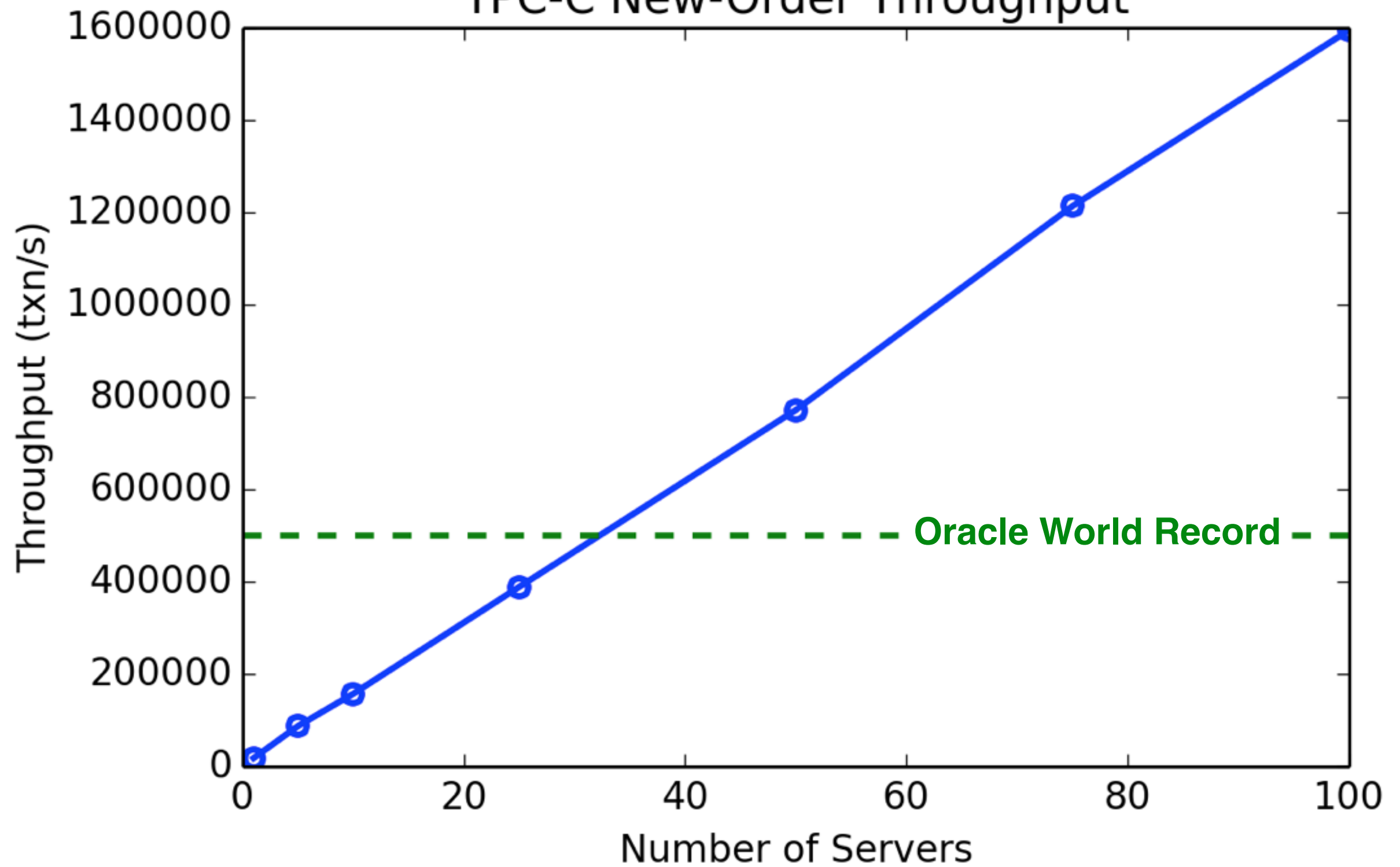
TPC-C New-Order Throughput



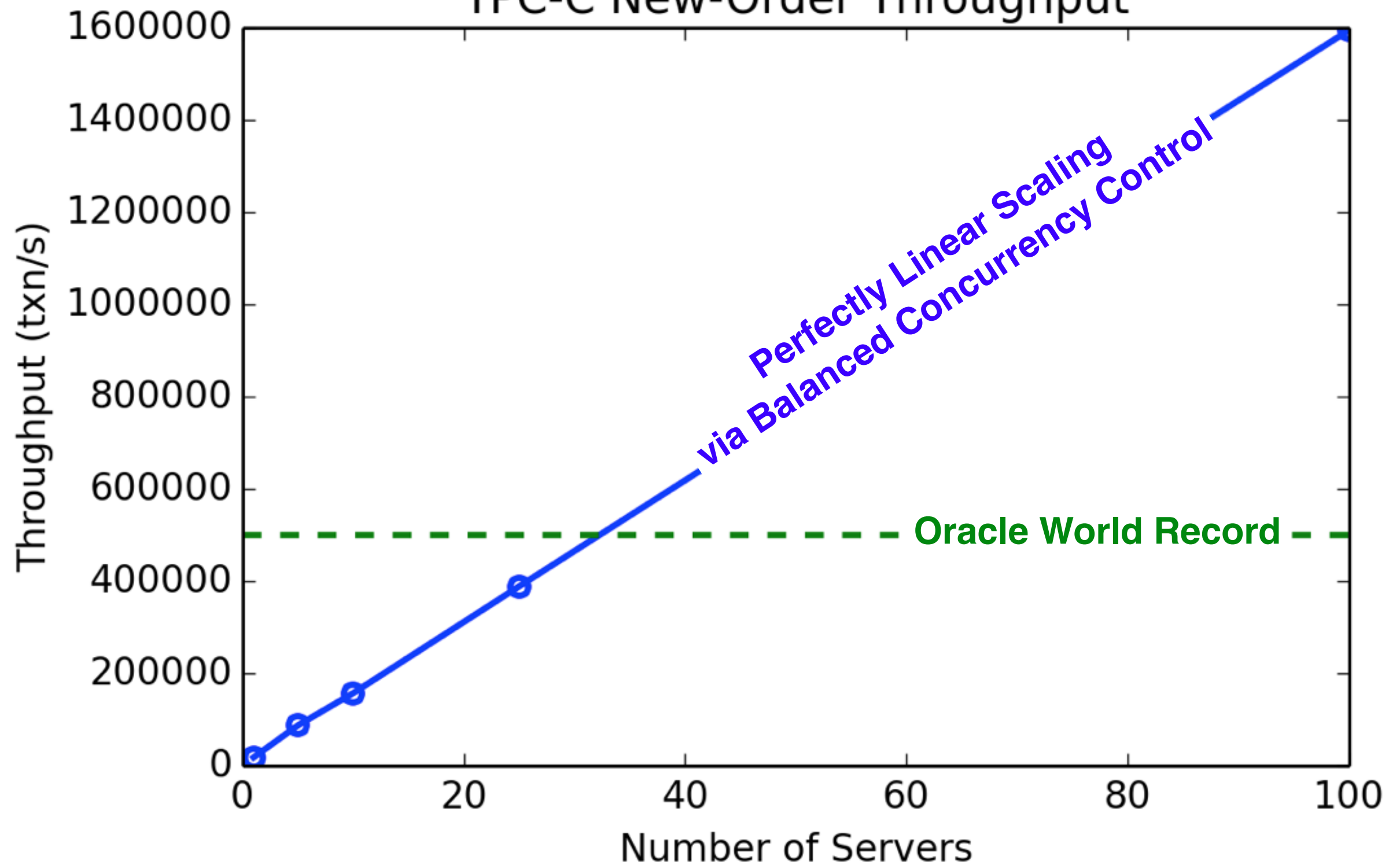
TPC-C New-Order Throughput



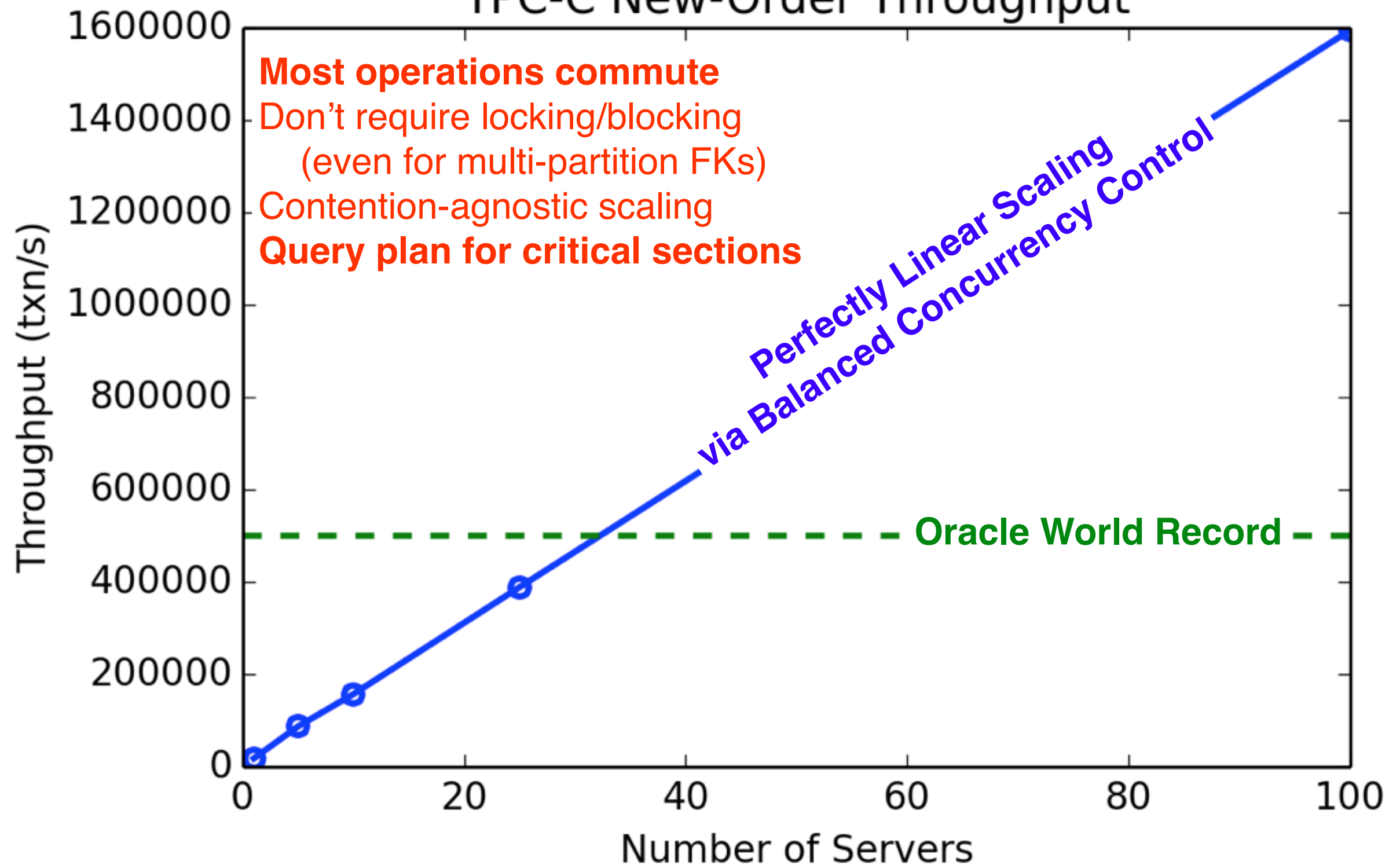
TPC-C New-Order Throughput



TPC-C New-Order Throughput



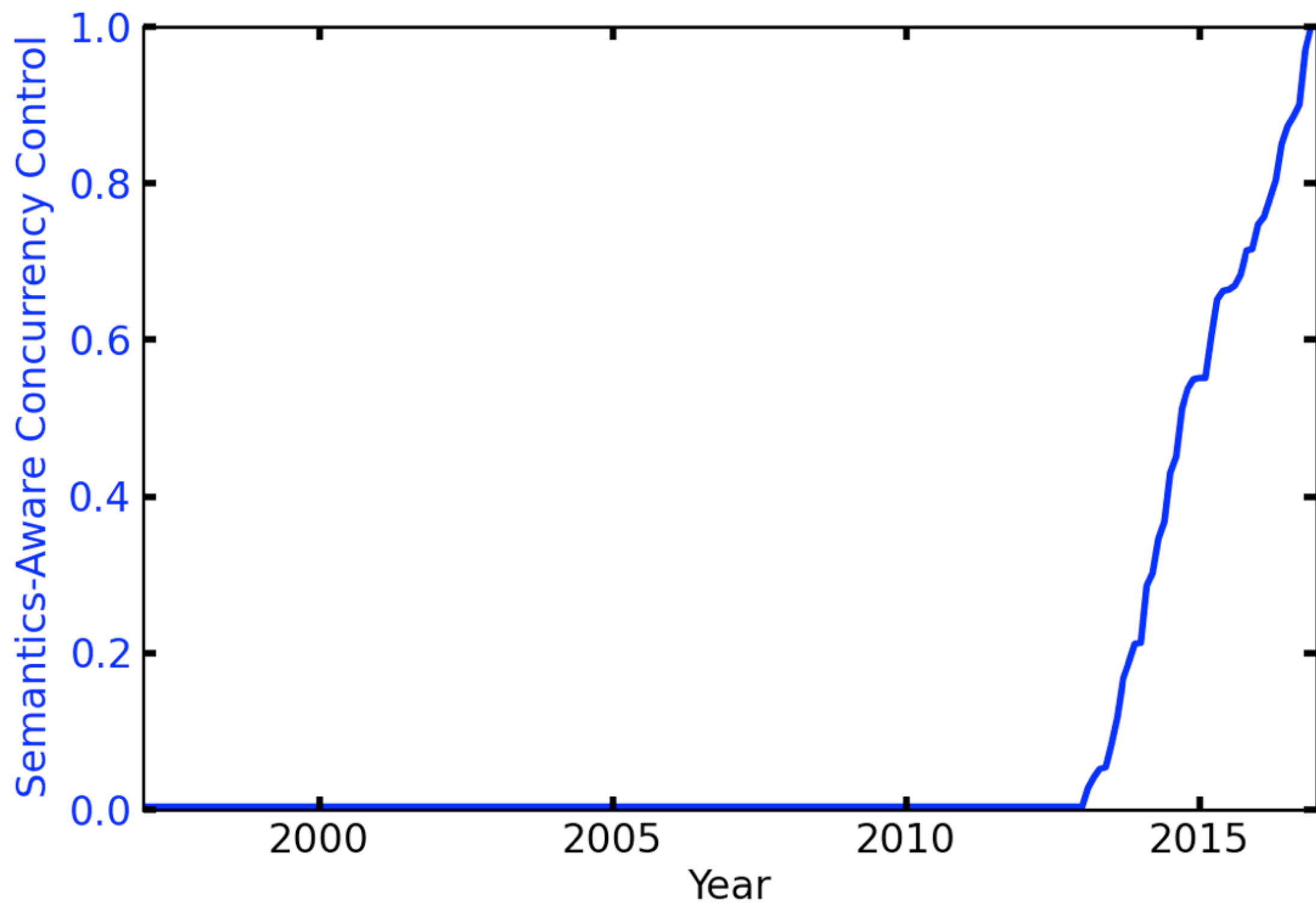
TPC-C New-Order Throughput

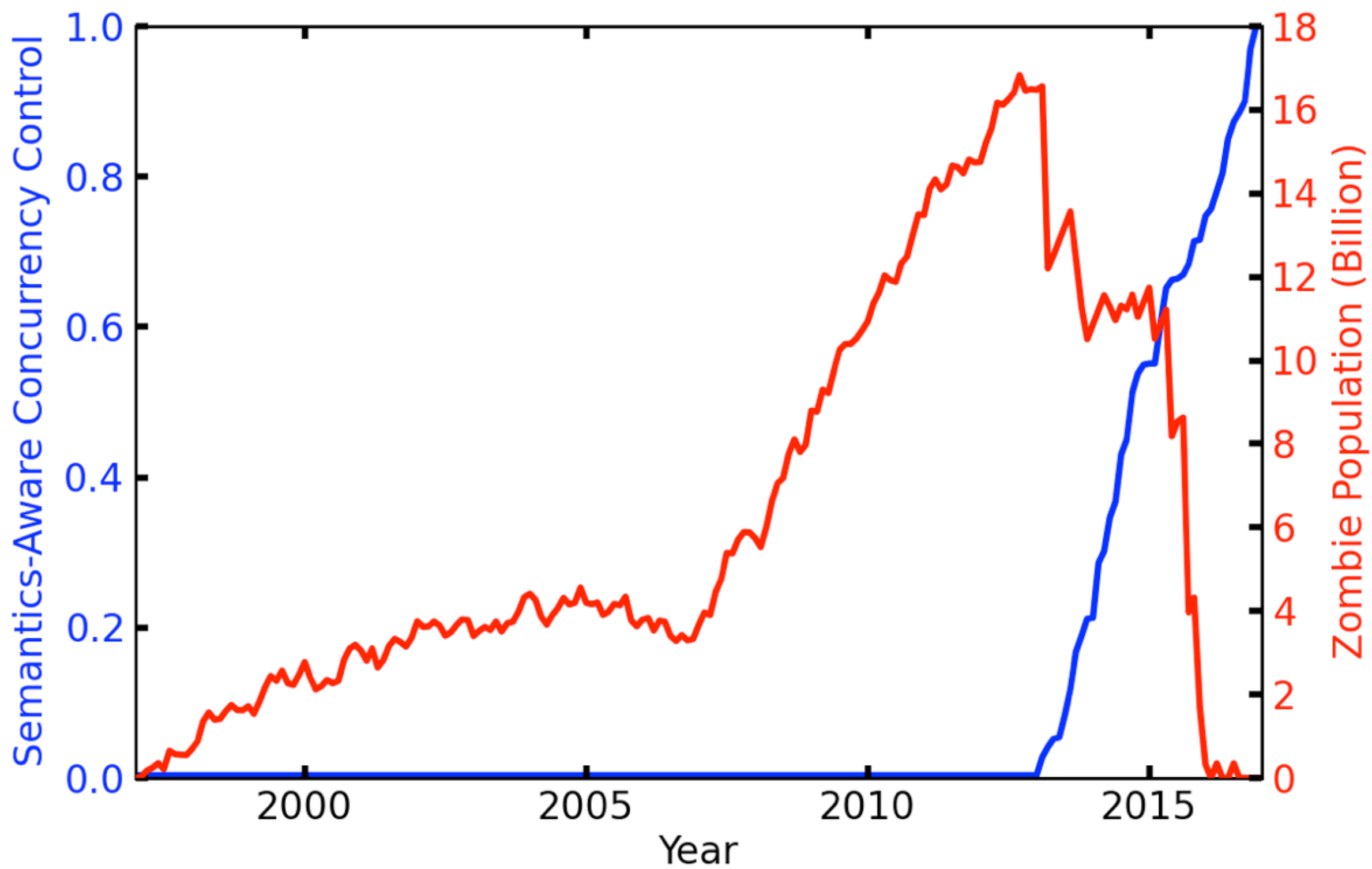


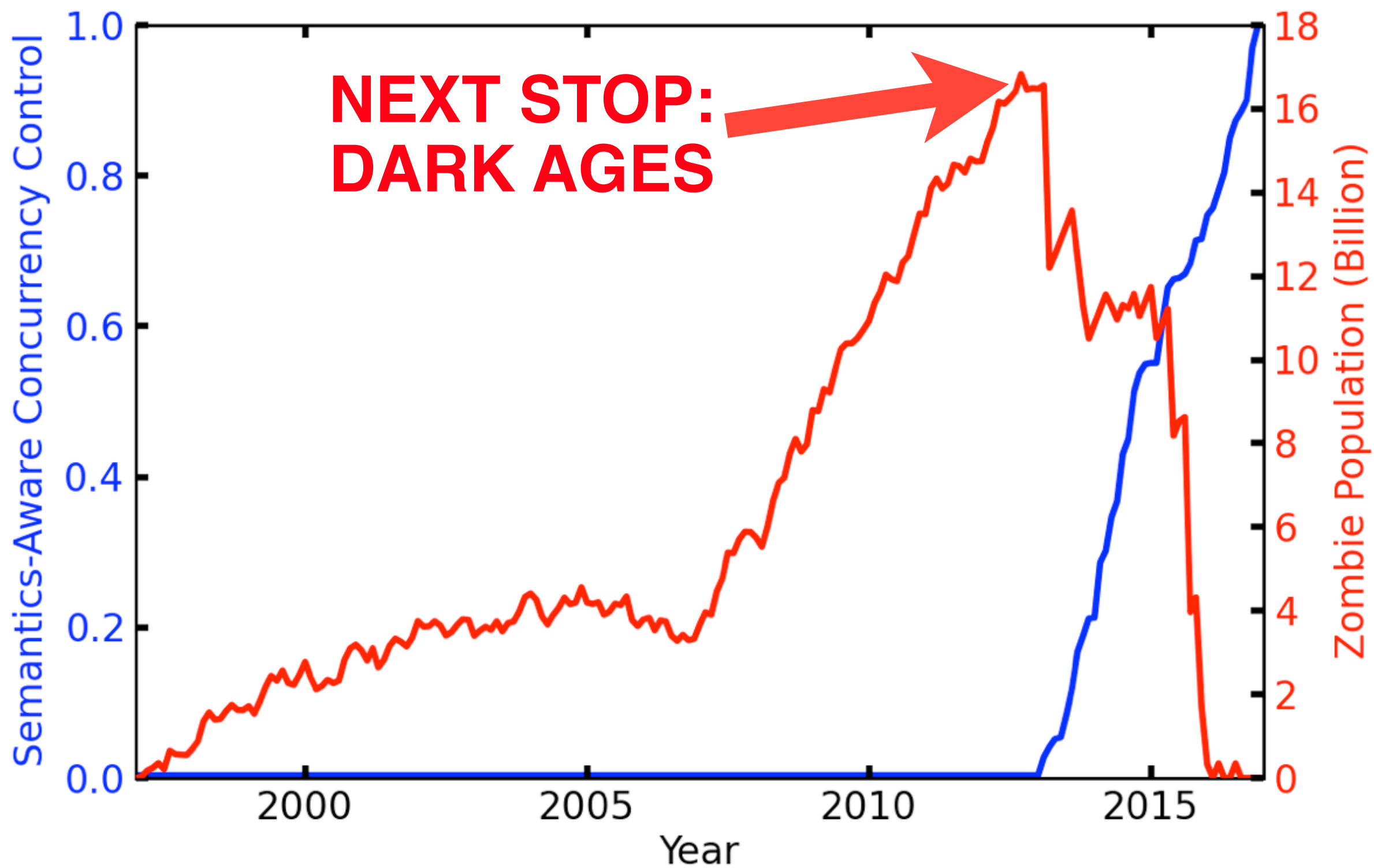
**Some results
went here**

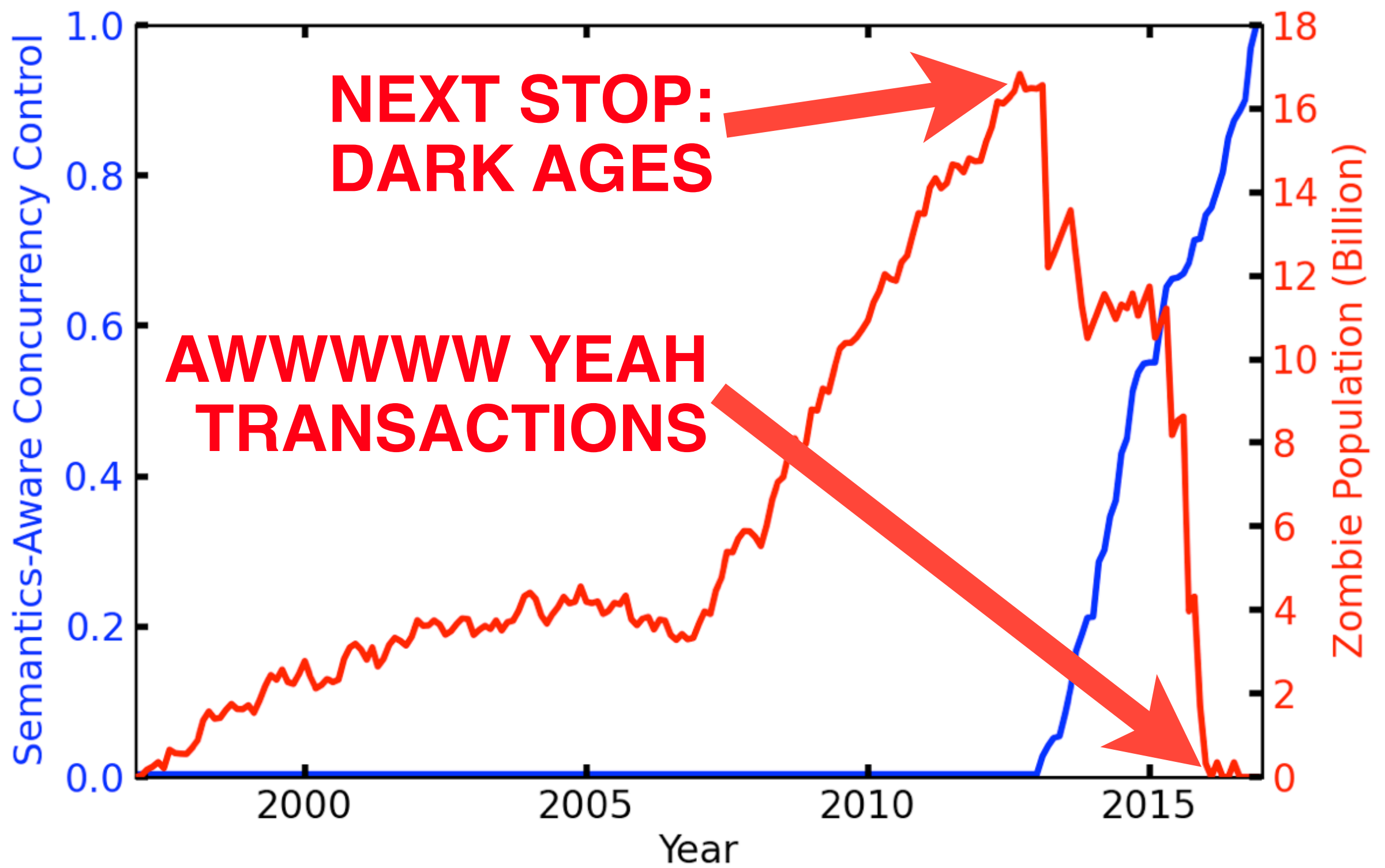
**WEBSCALE
ACID TXNS?
YES PLEASE!**











What changed?

What changed?

Took a broader view of database consistency:
application-level concerns take center stage

What changed?

Took a broader view of database consistency:
application-level concerns take center stage

Concurrency-control-aware query planning
(overdue: RC isolation no easier to comprehend than eventual consistency)

What changed?

Took a broader view of database consistency:
application-level concerns take center stage

Concurrency-control-aware query planning
(overdue: RC isolation no easier to comprehend than eventual consistency)

Continued maturation of NoSQL stores
RDBMS adoption of fast, replicated primitives

What changed?

Took a broader view of database consistency:
application-level concerns take center stage

Concurrency-control-aware query planning
(overdue: RC isolation no easier to comprehend than eventual consistency)

Continued maturation of NoSQL stores
RDBMS adoption of fast, replicated primitives

Research community engagement with industry
*(e.g, heeded scalability warnings,
Google released Spanner/F1 workload, benchmark results in 2014)*

2013:

2013:

Meet NoSQL on their turf

*(one isolation level does **not** fit all)*

Beat them at their own game

(scalability, via modern algorithms)

Restore the glory to database systems

(query planning ftw;

RDBMS is not a dirty word)

2013:

Meet NoSQL on their turf

*(one isolation level does **not** fit all)*

Beat them at their own game

(scalability, via modern algorithms)

Restore the glory to database systems

(query planning ftw;

RDBMS is not a dirty word)

2023:

2013:

Meet NoSQL on their turf

*(one isolation level does **not** fit all)*

Beat them at their own game

(scalability, via modern algorithms)

Restore the glory to database systems

(query planning ftw;

RDBMS is not a dirty word)

2023:

Still **no sequel** to the
zombie pandemic...

