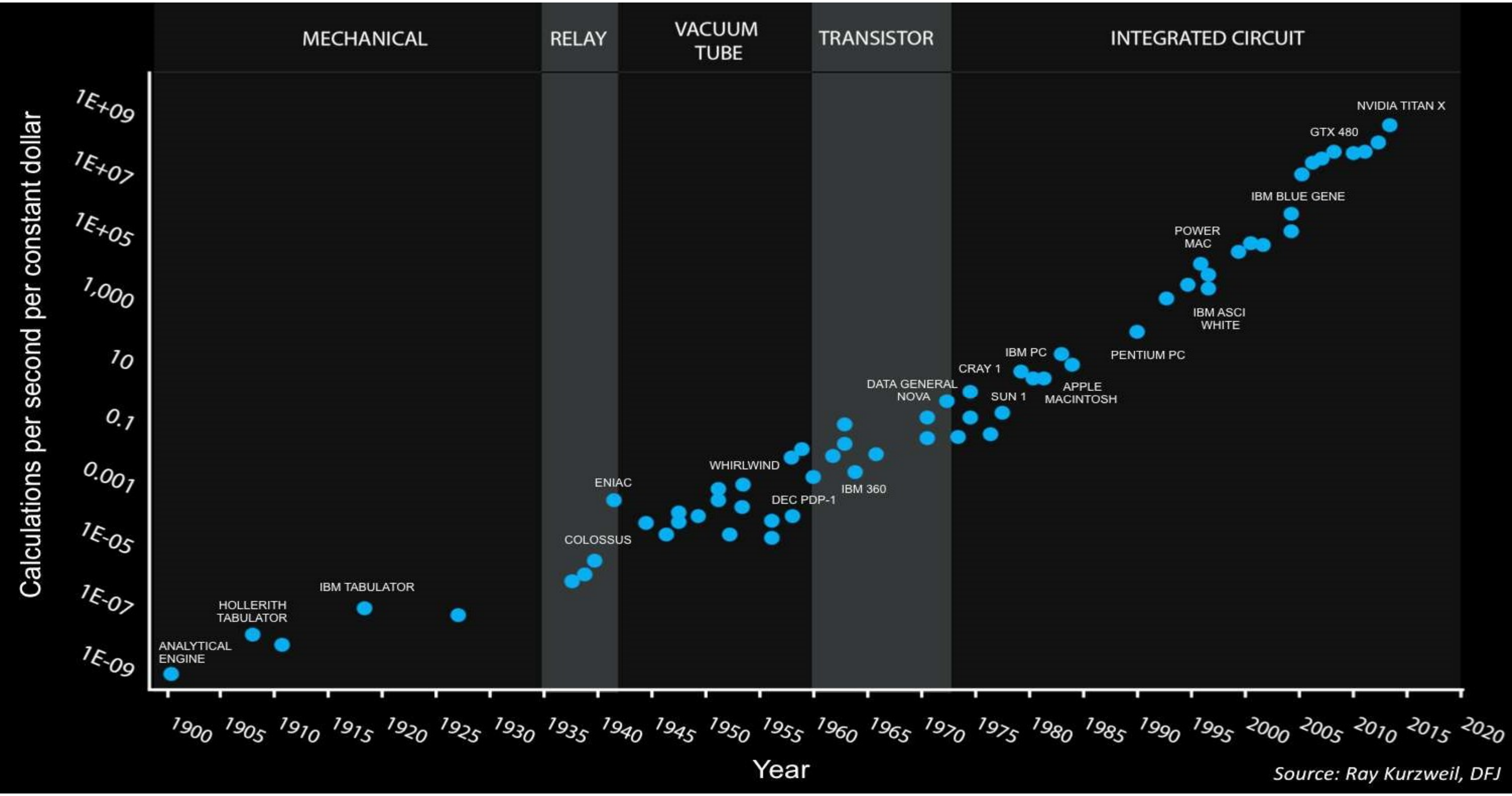
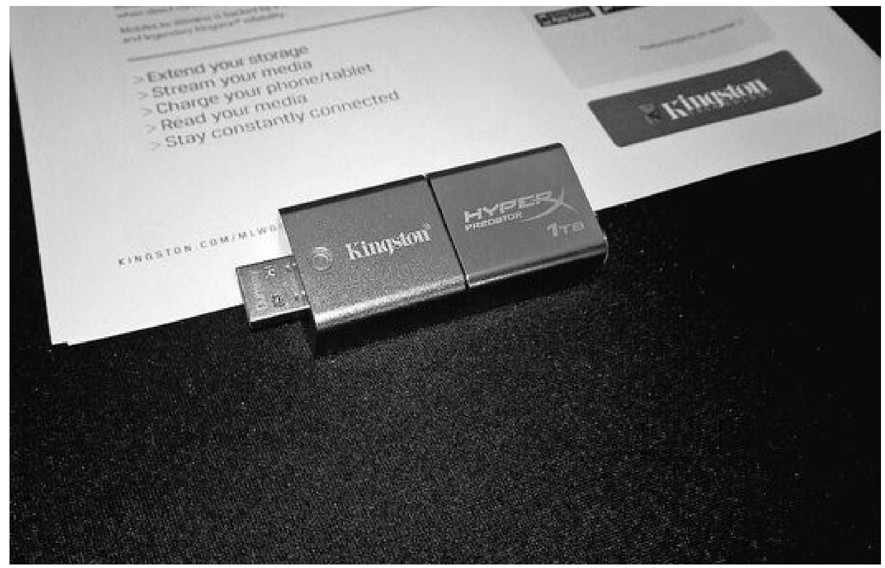
Concurrency in the Cloud

Johannes Gehrke

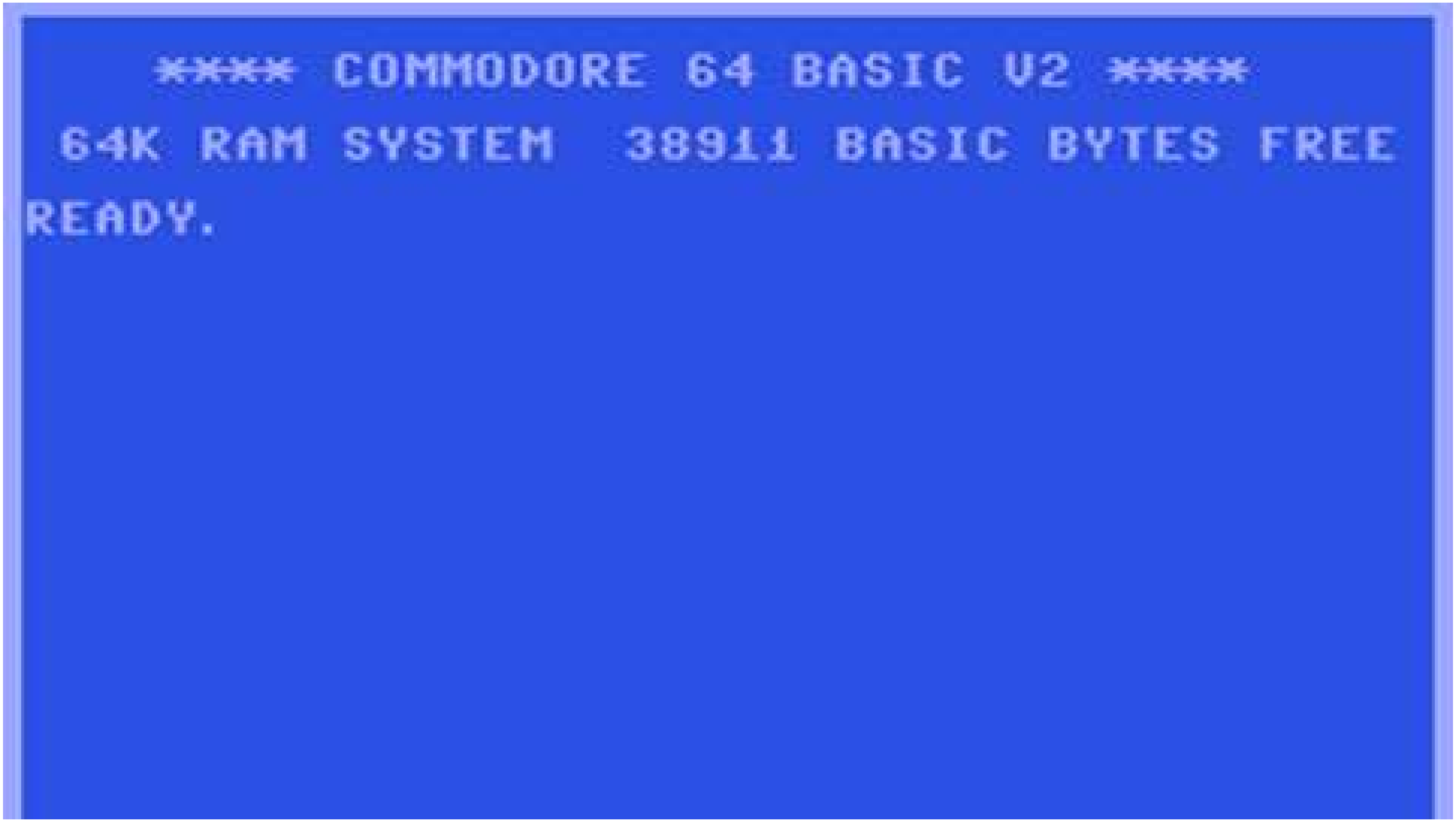
Microsoft Office Product Group

johannes@microsoft.com


MECHANICAL
RELAY
VACUUM TUBE
TRANSISTOR
INTEGRATED CIRCUIT
Calculations per second per constant dollar
1E+09
1E+07
1E+05
1,000
10
0.1
0.001
1E-05
1E-07
1E-09
NVIDIA TITAN X
GTX 480
IBM BLUE GENE
POWER MAC
IBM ASCI WHITE
PENTIUM PC
IBM PC
APPLE MACINTOSH
CRAY 1
SUN 1
DATA GENERAL NOVA
IBM 360
DEC PDP-1
WHIRLWIND
ENIAC
COLOSSUS
IBM TABULATOR
HOLLERITH TABULATOR
ANALYTICAL ENGINE
1900 1905 1910 1915 1920 1925 1930 1935 1940 1945 1950 1955 1960 1965 1970 1975 1980 1985 1990 1995 2000 2005 2010 2015 2020
Year
Source: Ray Kurzweil, DFJ

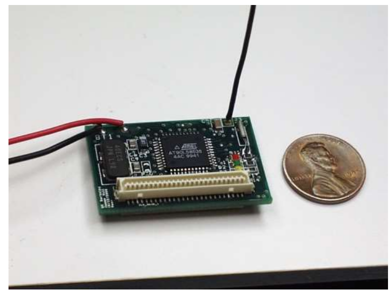


https://twitter.com/ValaAfshar



https://bits.blogs.nytimes.com/2011/04/06/the-new-commodore-64-updated-with-its-old-exterior/

A small Hardware Revolution



- Moore's Law
  - In 1965, Intel Corp. cofounder Gordon Moore predicted that the density of transistors in an integrated circuit would double every year.
  - Later changed to reflect 18 months progress.

http://lecs.cs.ucla.edu/Resources/testbed/testbed-overview.html

A small Hardware Revolution

- Experts on ants estimate that there are 10^{16} to 10^{17} ants on earth. In the year 1997, we produced one transistor per ant.
  [Gordon Moore, http://www.intel.com/pressroom/archive/speeches/gem93097.htm]

Image from https://www.123rf.com/photo_14232419_the-big-ant-hill-in-a-woods.html

One Example: The Microsoft Cloud

https://blogs.msdn.microsoft.com/uk_faculty_connection/2016/09/19/azure-data-centers-and-regions/

What is Interesting about the Cloud?

My days in academia:

- Scalability
- Elasticity
- Multi-tenancy
- Infrastructure, new abstractions
- Resource management
- Security
- Energy
- …

Now in industry in addition:

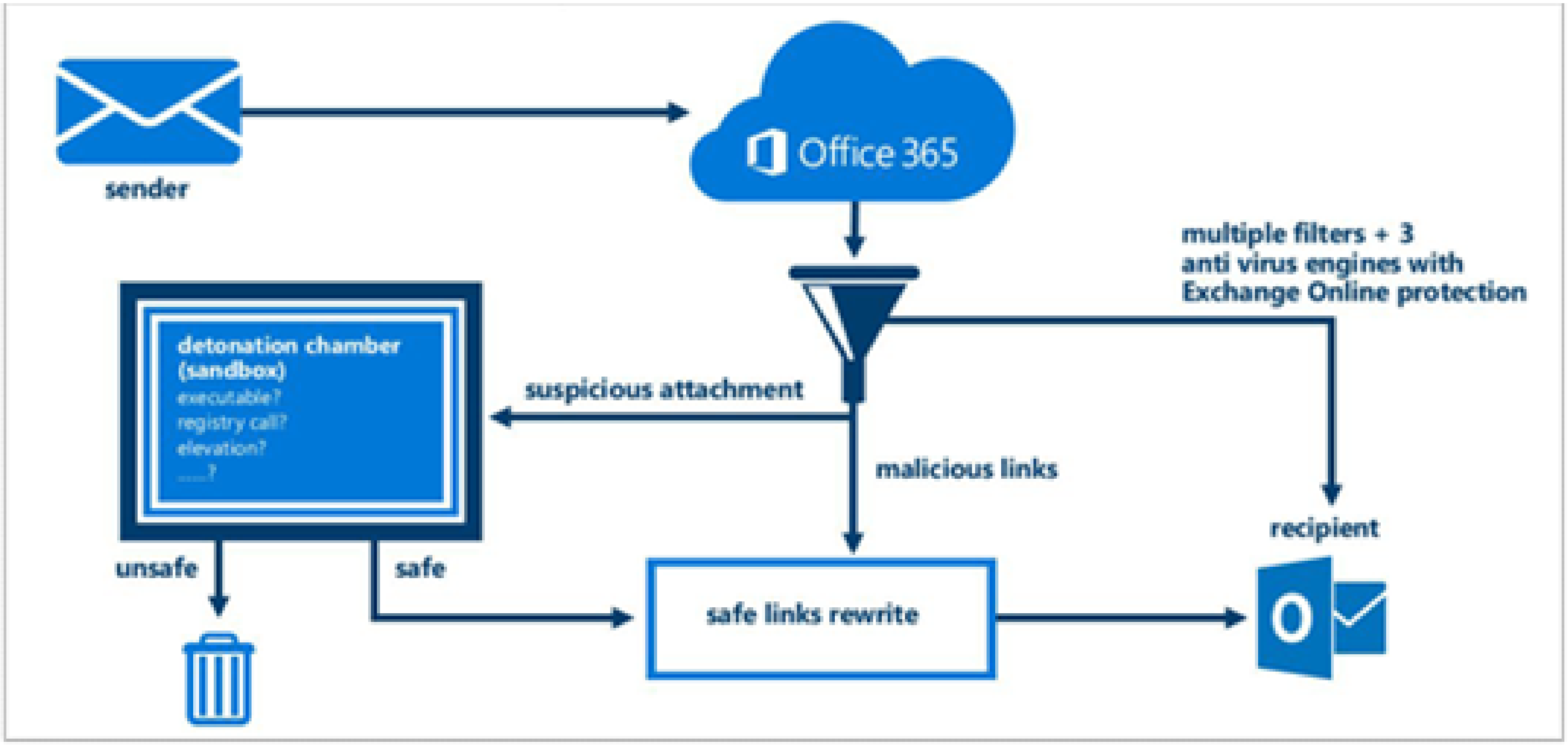
- Keeping up with growth/scale
- Running the service
- Innovation across the stack, from hardware to software
- Machine learning
- Practical security
- Storage

New Hardware

- Compute
- Network
- Memory

https://www.microsoft.com/en-us/research/project/project-catapult/

Example: Office 365 Security



https://products.office.com/en-us/exchange/online-email-threat-protection

Storage

Canada East
Canada Central
US Gov Iowa
Central US
West US 2
West US
US Gov Arizona
West Central US
US Gov Texas
South Central US
North Central US
US DoD East
East US
East US 2
US Gov Virginia
US DoD Central
UK South
North Europe
UK West
West Europe
Germany Northeast
Germany Central
France Central
France South
Korea Central
Korea South
Japan East
Japan West
China North
China East
East Asia
West India
Central India
South India
Southeast Asia
Brazil South
Australia East
Australia Southeast
Generally available
Coming soon

One Example: Azure DocumentDB

- Global availability
  - Navigate CAP theorem
  - Single-system image of any table, across all datacenters
  - Physical realities such as the speed of light matter
- Automatic multi-region replication
  - Automatic partition management
  - Associate any number of regions with your database account
  - Policy based geo-fencing
- Multi-homing APIs
  - Apps don't need to be redeployed during regional failover
- Offers comprehensive SLA that includes latency, throughput, availability and consistency
- https://azure.microsoft.com/en-us/services/documentdb/

Slide based on material from Rimma Nehme and Dharma Shukla

Guaranteed Low Latency

	Reads (1KB)	Indexed writes (1KB)
50th	<2ms	<6ms
99th	<10ms	<15ms

- Globally distributed with reads and writes served from local region
  - Write optimized, latch-free database engine designed for SSDs and low latency access
  - Synchronous and automatic indexing at sustained ingestion rates

Slide courtesy of Rimma Nehme and Dharma Shukla

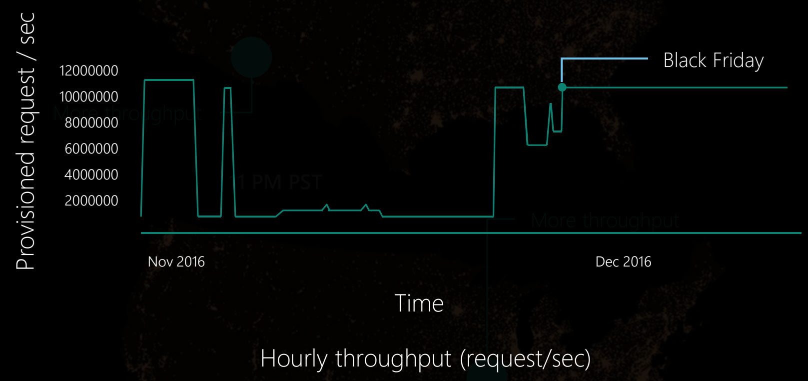
Elastically Scalable Throughput

Elastically scale throughput from 10 to 100s of millions of requests/sec across multiple regions

Customers pay by the hour for the provisioned throughput.

Transparent server side partition management and routing

Support for requests/sec and requests/min for different workloads



Slide courtesy of Rimma Nehme and Dharma Shukla

The 99.99 SLA

Version History
1.1 Last Updated: March 2017
Improved SLA to include guarantees for consistency, latency and throughput, in addition to availability.
Expanded the definition of Database Account, removed the Excluded Requests definition and modified the Total Requests definition.

1.0 Last Updated: March 2016
Revised to reflect increment to 99.99% monthly uptime percentage

SLA for DocumentDB

Last updated: March 2017

Azure DocumentDB is Microsoft's multi-tenant, globally distributed database system designed for the cloud. DocumentDB allows customers to provision and elastically scale throughput and storage across any number of geographical regions. The service offers 99.99% guarantees for availability, throughput, low latency, and consistency.

SLA Details

In the event that more than one Service Level for Azure DocumentDB are not met, you must choose only one Service Level under which to make a claim for your Service Credit for Azure DocumentDB. Service Credits for different Service Levels for Azure DocumentDB may not be combined in a given month. You are entitled only to the Service Credit for the particular Service Level under which you make a claim.

Additional Definitions

"**Database Account**" is the top-level resource of the DocumentDB resource model. A DocumentDB Database Account contains one or more databases.

Slide courtesy of Rimma Nehme and Dharma Shukla

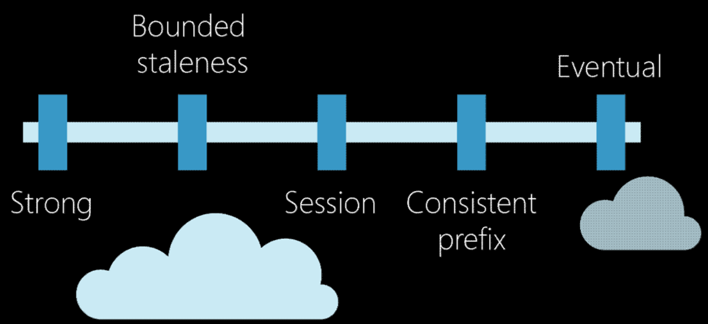
Well-Defined Consistency Models

Global distribution forces us to navigate the CAP theorem

Intuitive programming model for well-defined, relaxed consistency models

Five well-defined consistency levels to choose from

Can be overridden on a per request basis

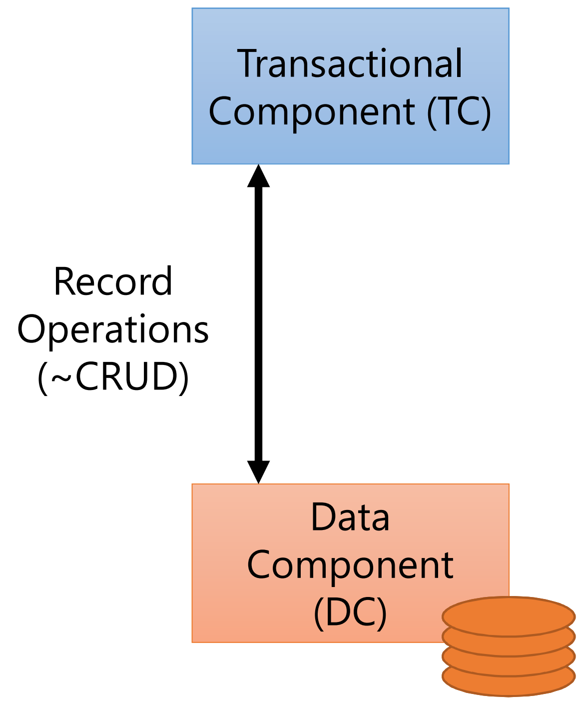


Slide courtesy of Rimma Nehme and Dharma Shukla

Under the Hood

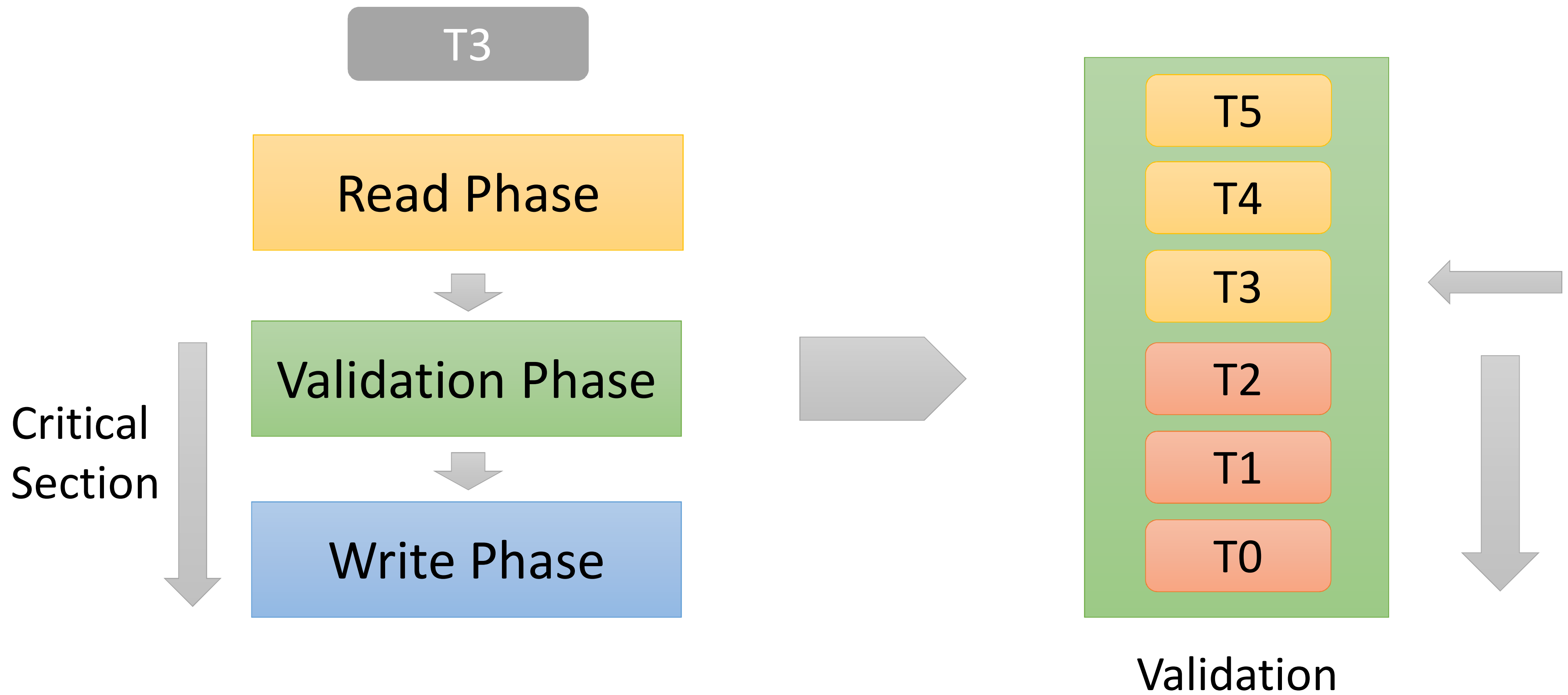
- TC guarantees ACID
  - Logical concurrency control
  - Logical recovery
  - No knowledge of physical data storage

- DC provides atomic record store
  - Physical data storage
  - Atomic record modifications
  - Must be logically consistent during operation
    - May require its own recovery



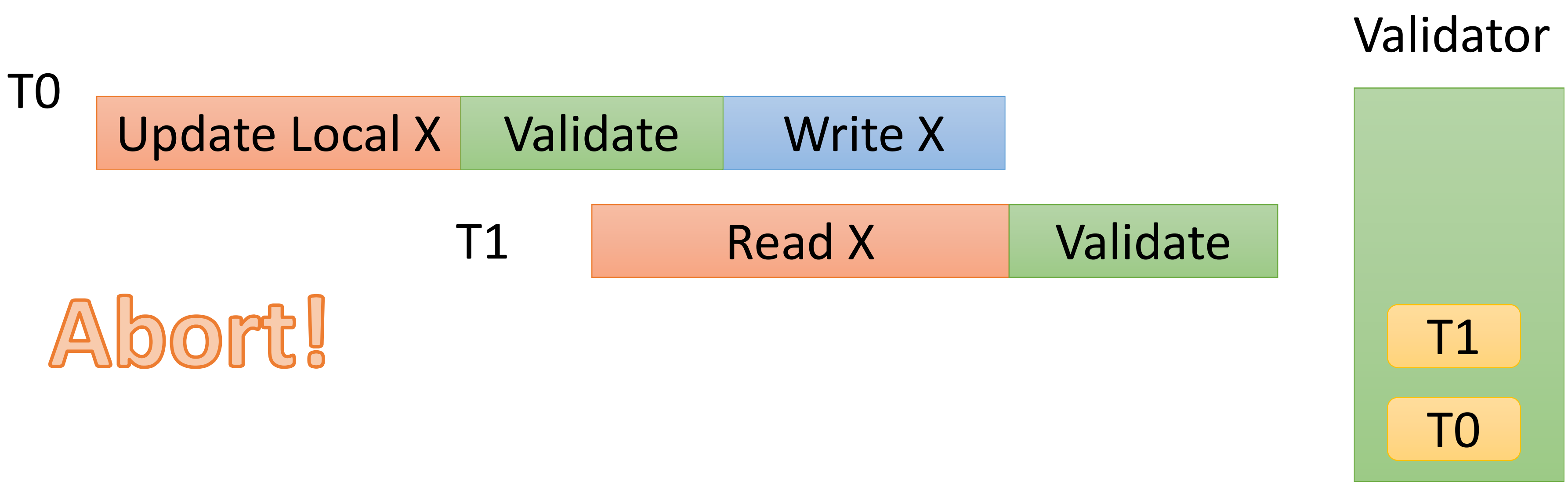
Slide adapted from David Lomet

Recall: Optimistic Concurrency Control



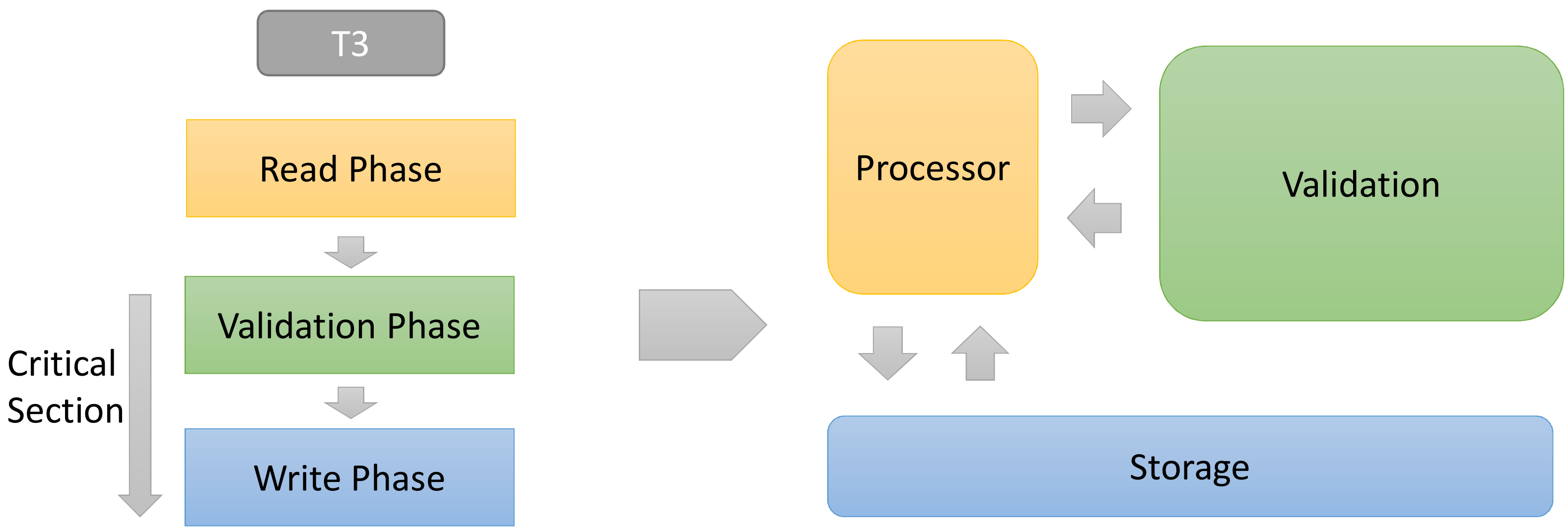
Validation in OCC

- Compare read set with all previously committed write sets



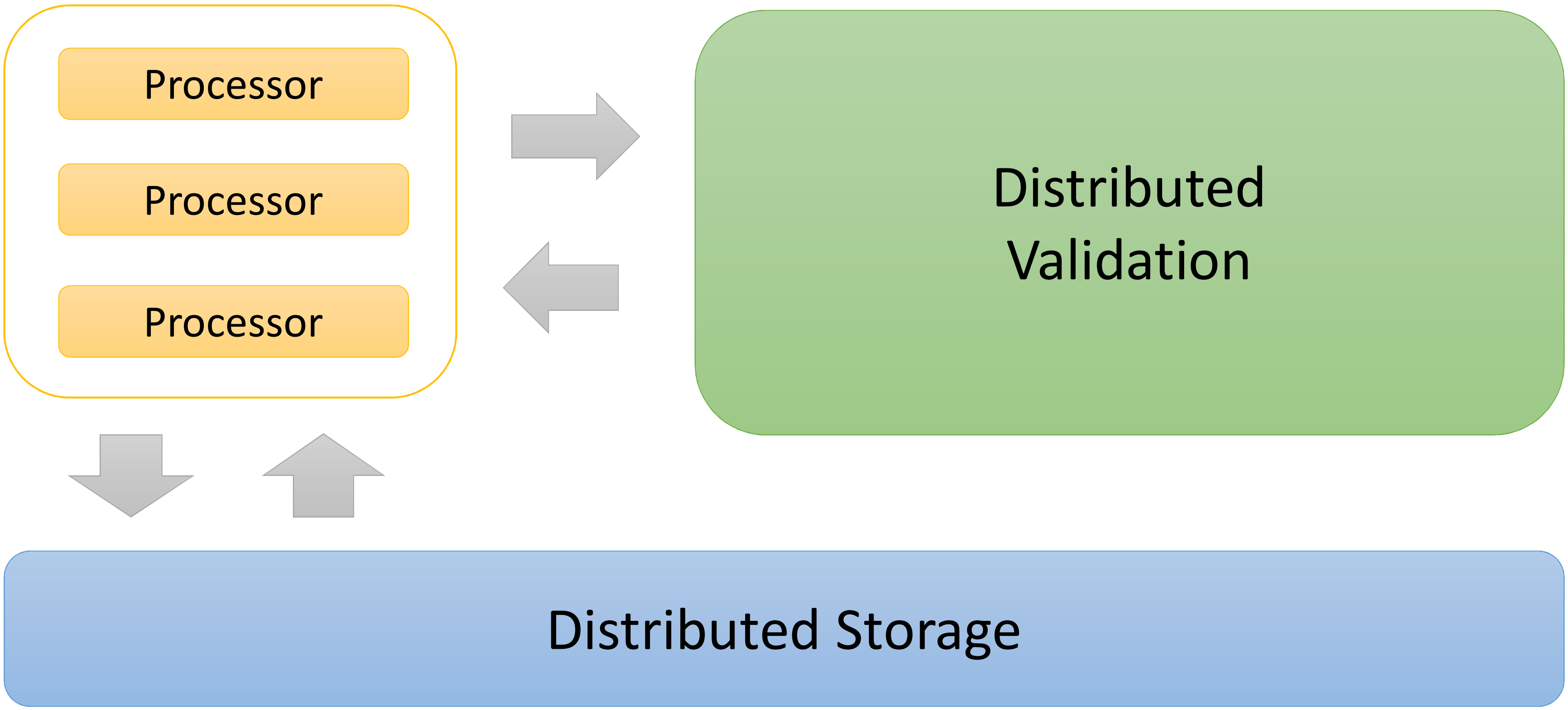
Optimistic Concurrency Control in Practice [1]

- Separation into functional components
- Design principle: simplicity and loose coupling

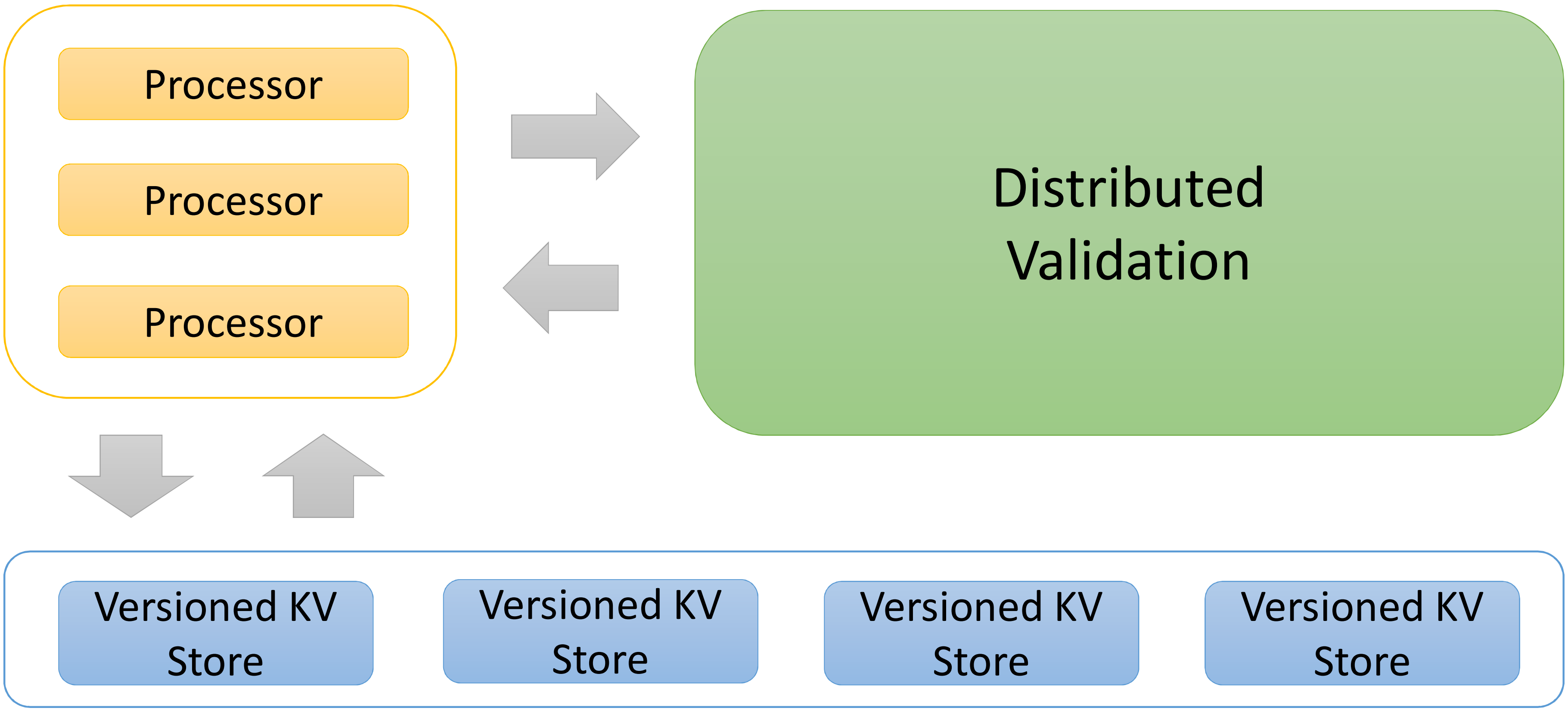


[1] Bailu Ding, Lucja Kot, Alan J. Demers, Johannes Gehrke: Centiman: elastic, high performance optimistic concurrency control by watermarking. SoCC 2015: 262-275

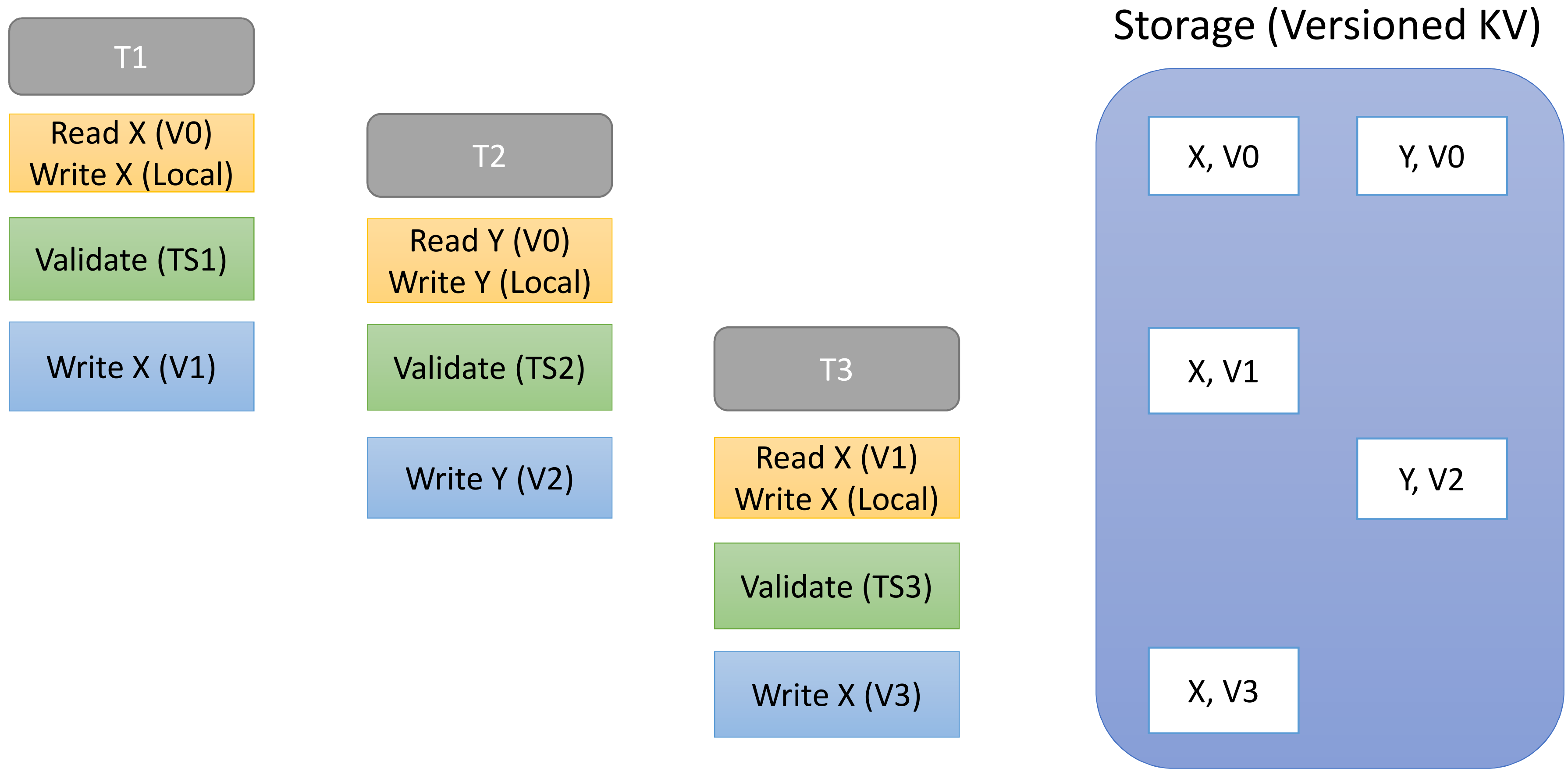
Scale-Out Processing



Scale-Out Storage

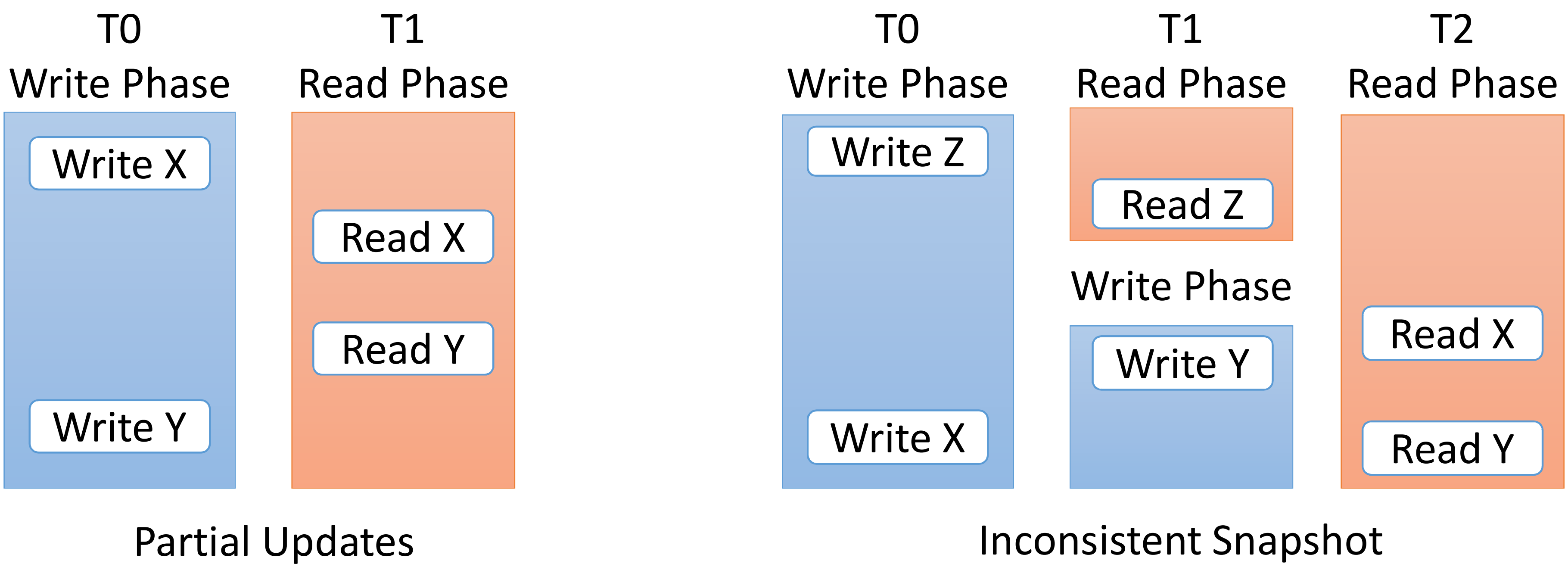


Multi-Versioned Storage

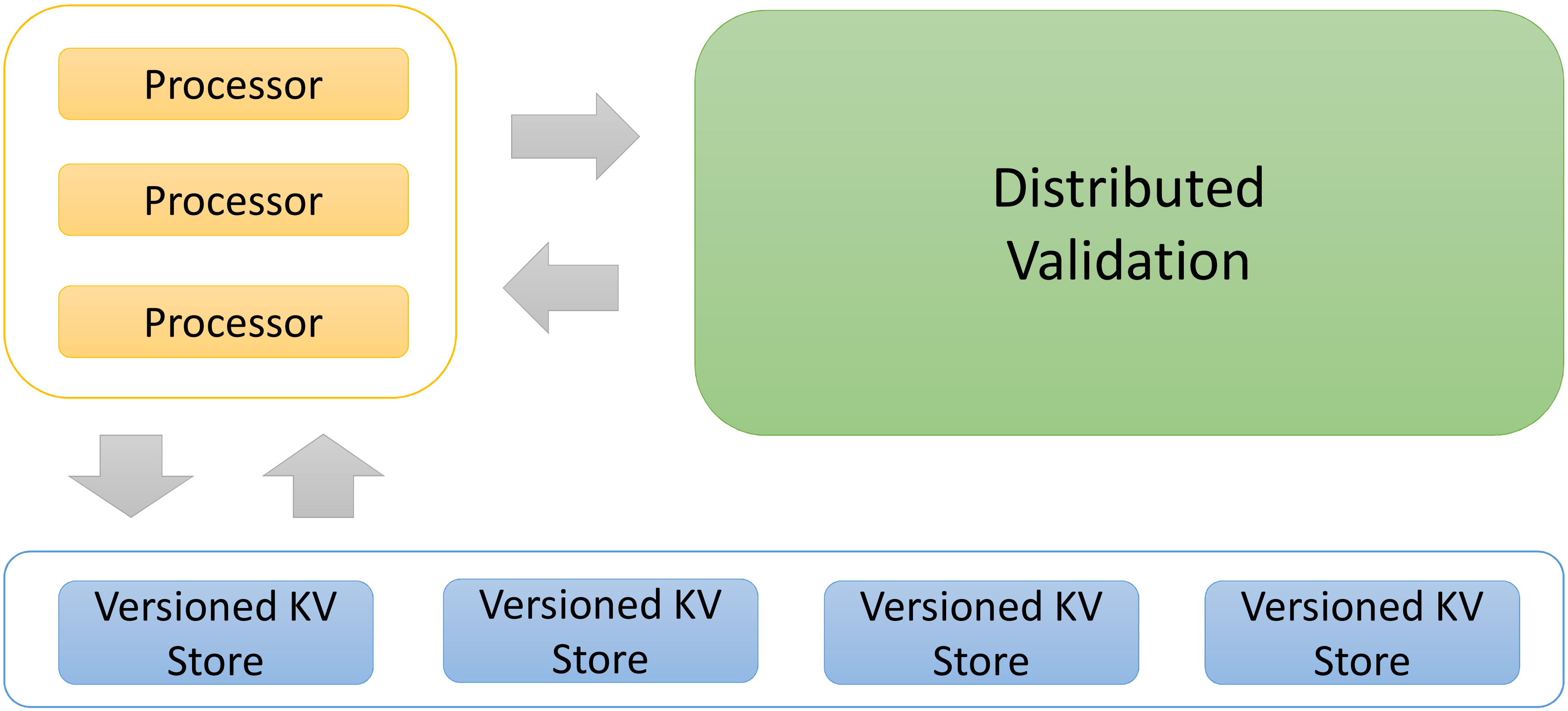


Reading Inconsistent Snapshots?

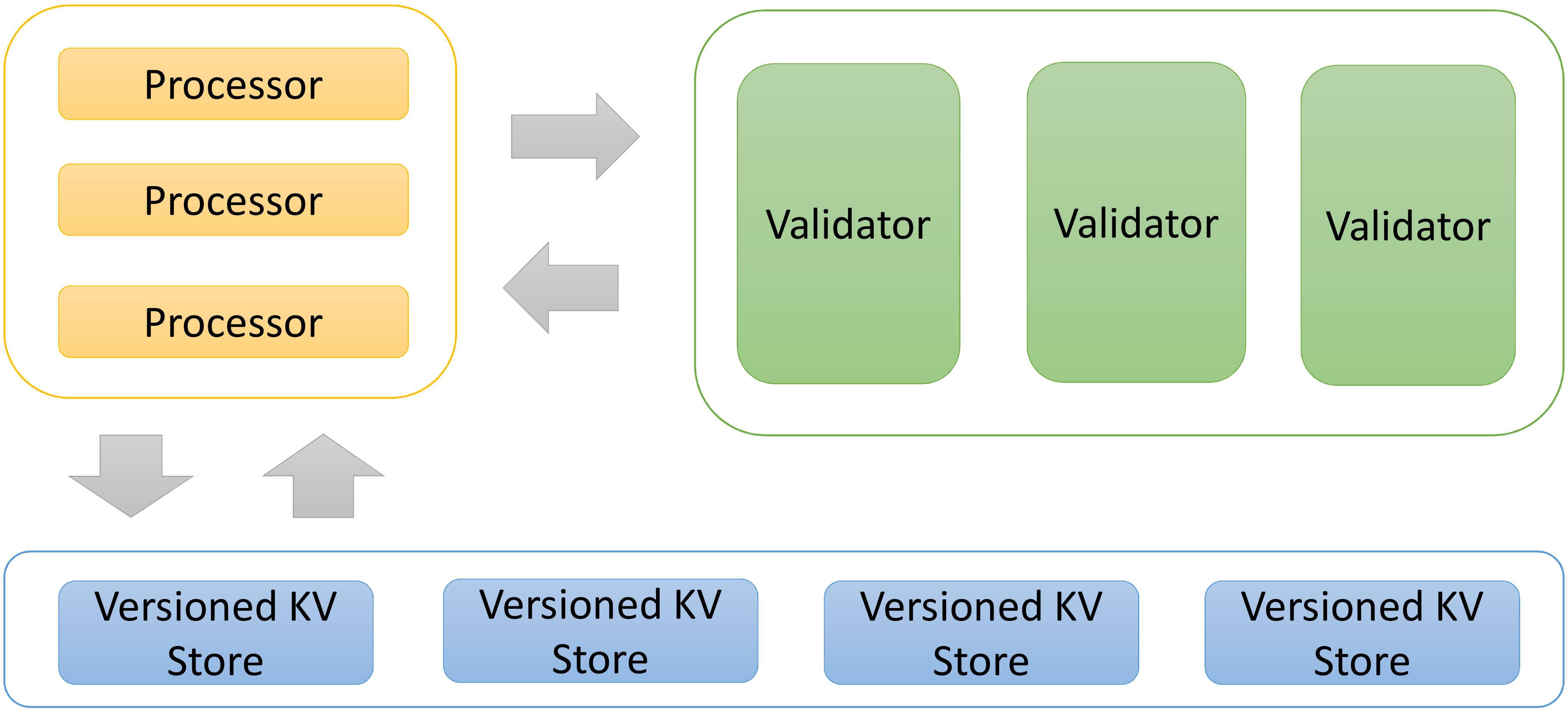
- Updates not installed atomically
- Approach: Check against inconsistent reads at validation



Scale-Out Validation



Scale-Out Validation (Contd.)

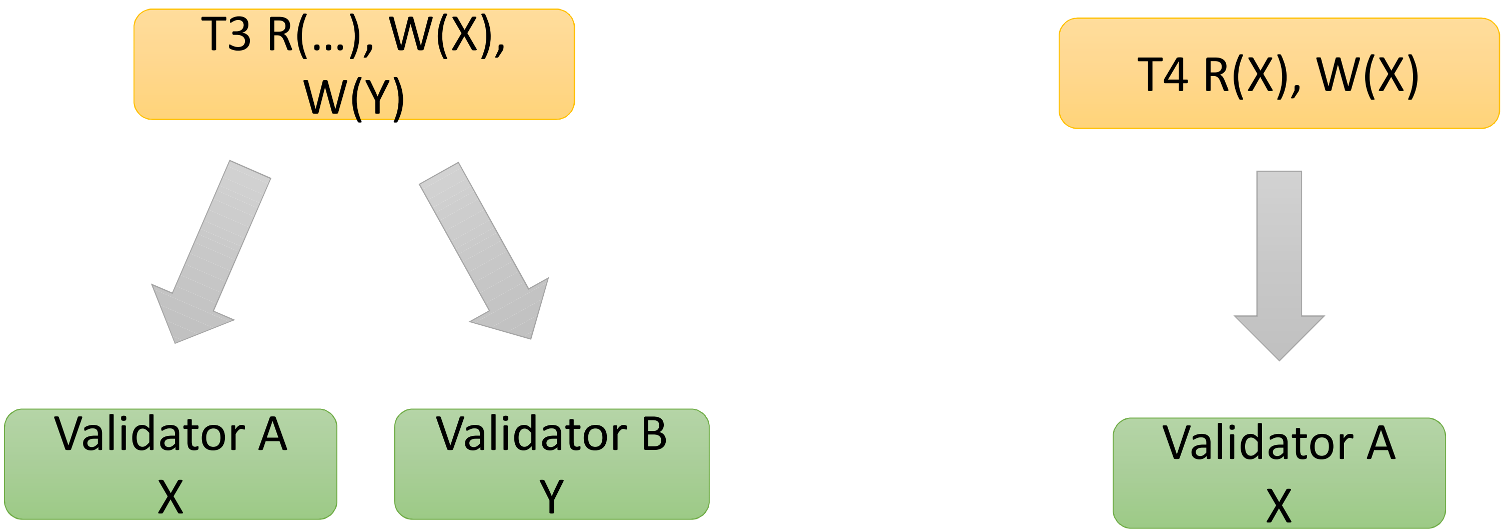


SD7
SD8
PB48

But: Divergent Decisions

Abort!

Spurious abort due to a spurious update!



No conflict!

Conflict: abort!

Thinks T3 committed, detects conflict between T3 and T4

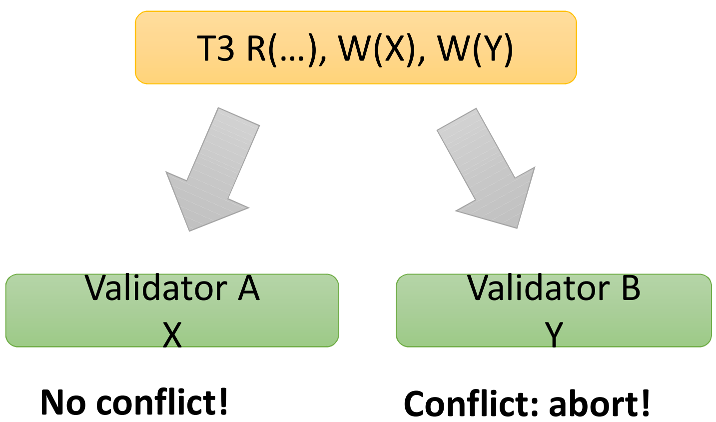
SD7 Why not use distributed OCC which runs a 2PC like validation algorithm? For instance: http://web.cs.ucdavis.edu/~wu/ecs251/ecs251_DMVOCC.pdf
Sudipto Das, 2/16/2016

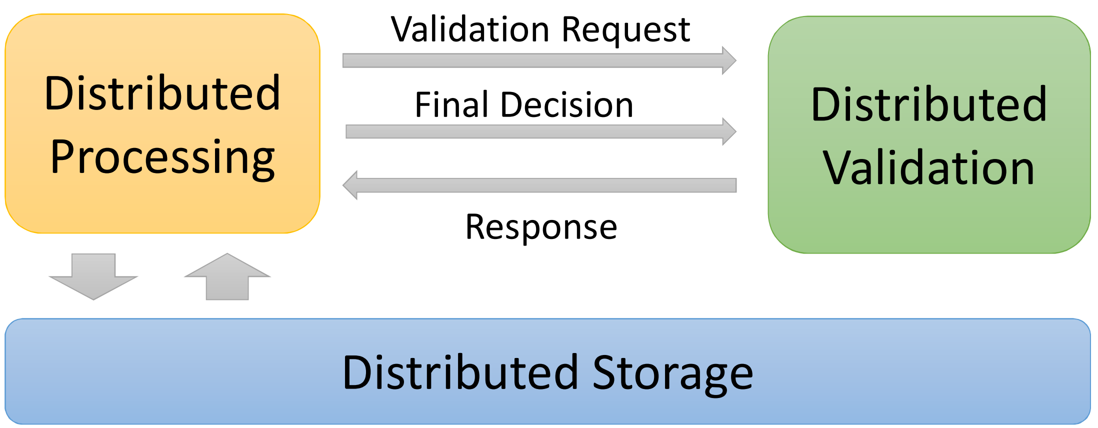
SD8 I think there are many variants of distributed OCC which does not have the problem of divergent decisions. While this is a good illustration, people will be quick to latch on that this is not the state-of-the-art.
Sudipto Das, 2/16/2016

PB48 Bailu, do you have an experimental result that shows the throughput of a 2PC-like distributed validation?
Phil Bernstein, 2/16/2016

Eliminate Spurious Aborts: Proactive

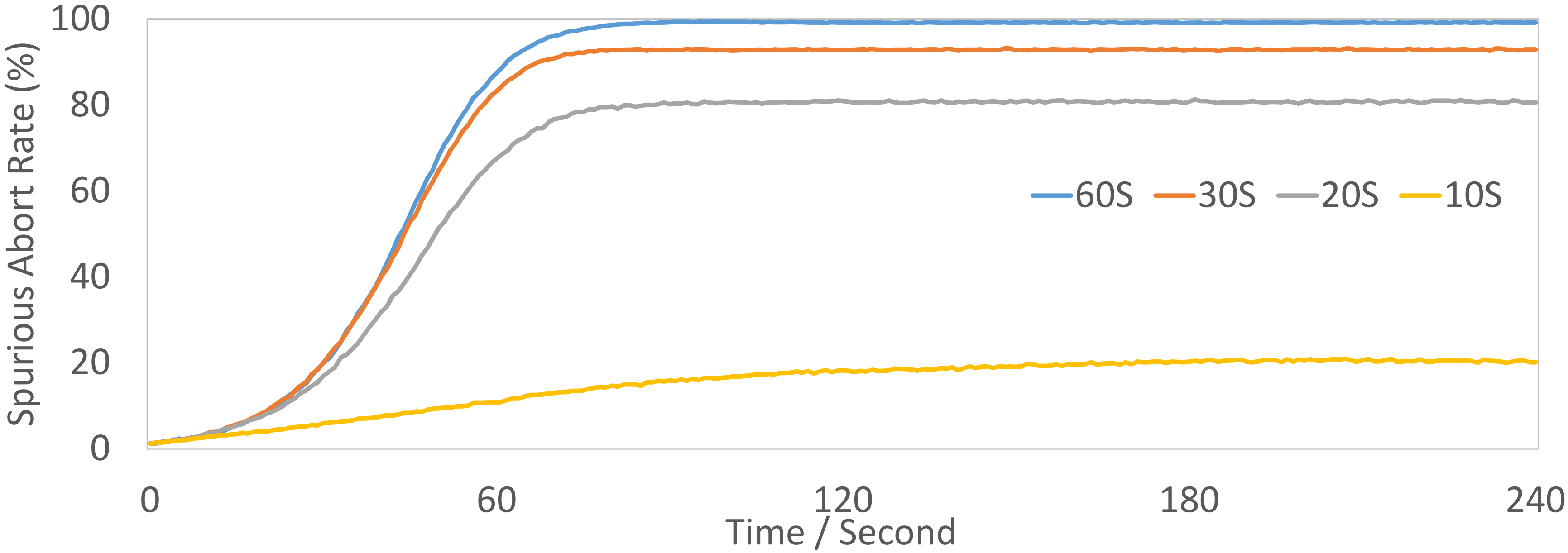
- Processor informs validators about the final decision
- Synchronous: slows down the system
- Asynchronous: Adds additional complexity since we need to revoke updates





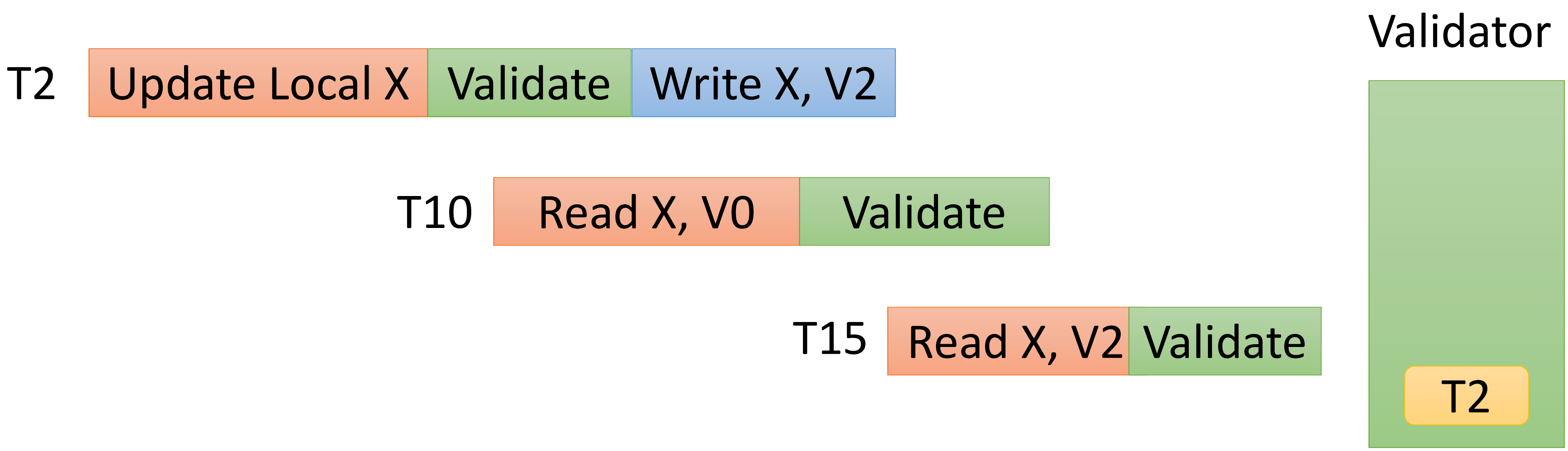
Eliminate Spurious Aborts: Lazy

- Old updates will be eventually discarded, i.e. garbage collection
- Reply on garbage collection to eliminate spurious updates



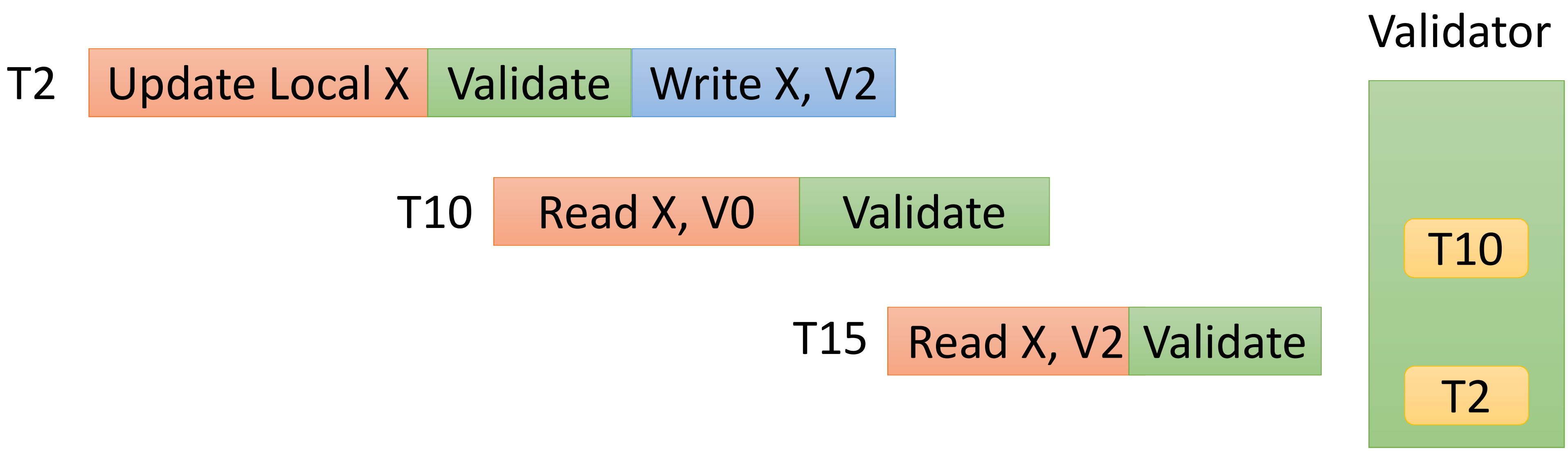
Eliminate Spurious Aborts: Lazy

- Lower the spurious abort rate by reducing the expiration time
- Risk of aggressive garbage collection: abort due to insufficient information



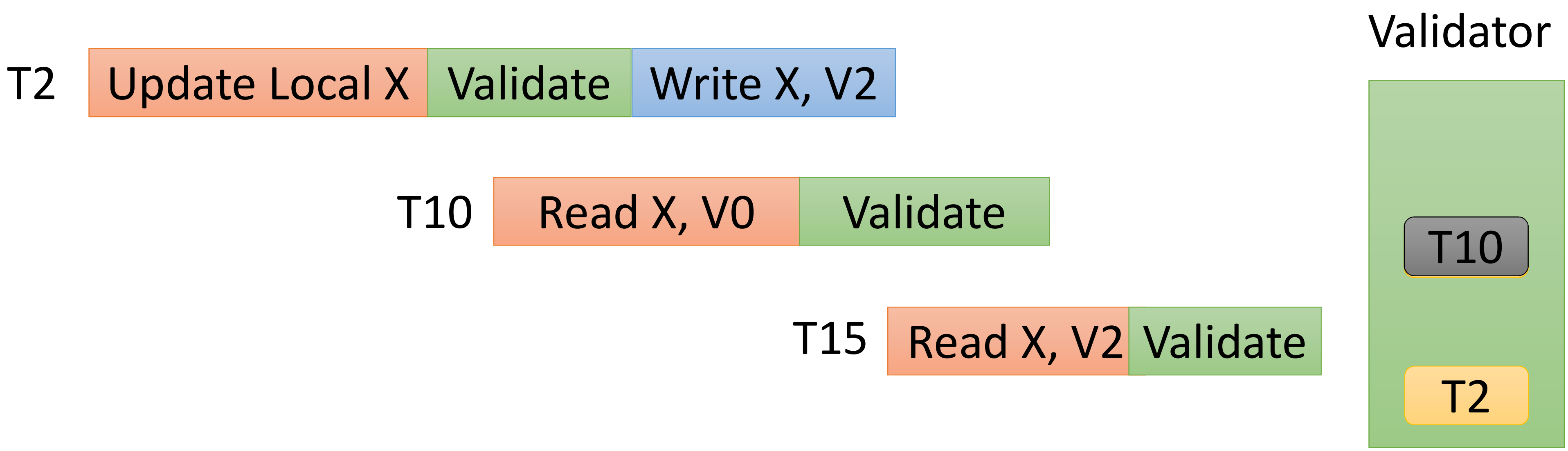
Eliminate Spurious Aborts: Lazy

- Lower the spurious abort rate by reducing the expiration time
- Risk of aggressive garbage collection: abort due to insufficient information



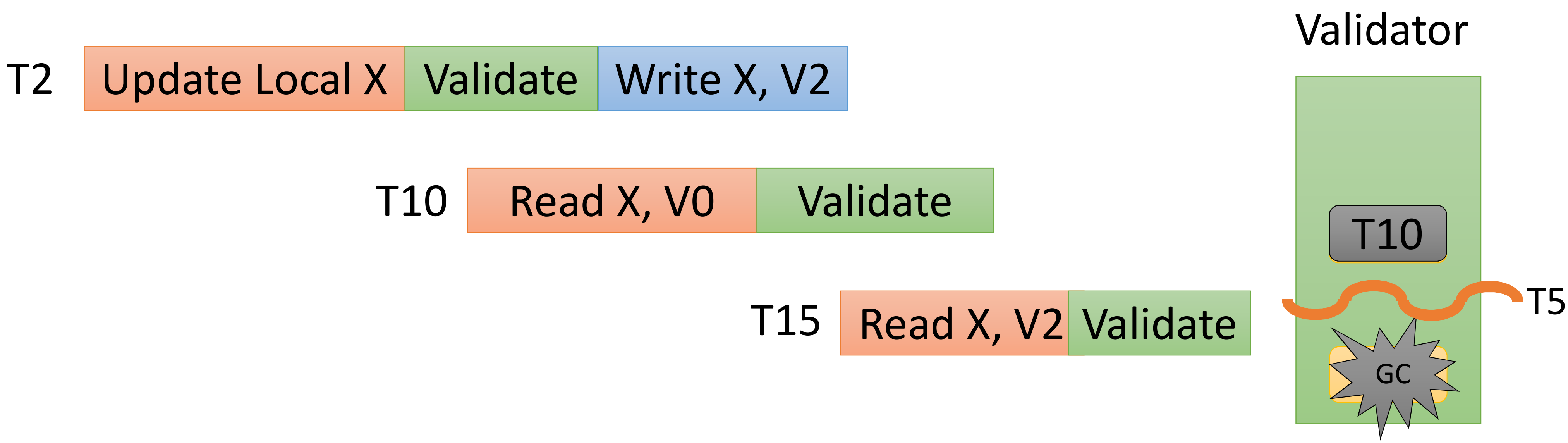
Eliminate Spurious Aborts: Lazy

- Lower the spurious abort rate by reducing the expiration time
- Risk of aggressive garbage collection: abort due to insufficient information



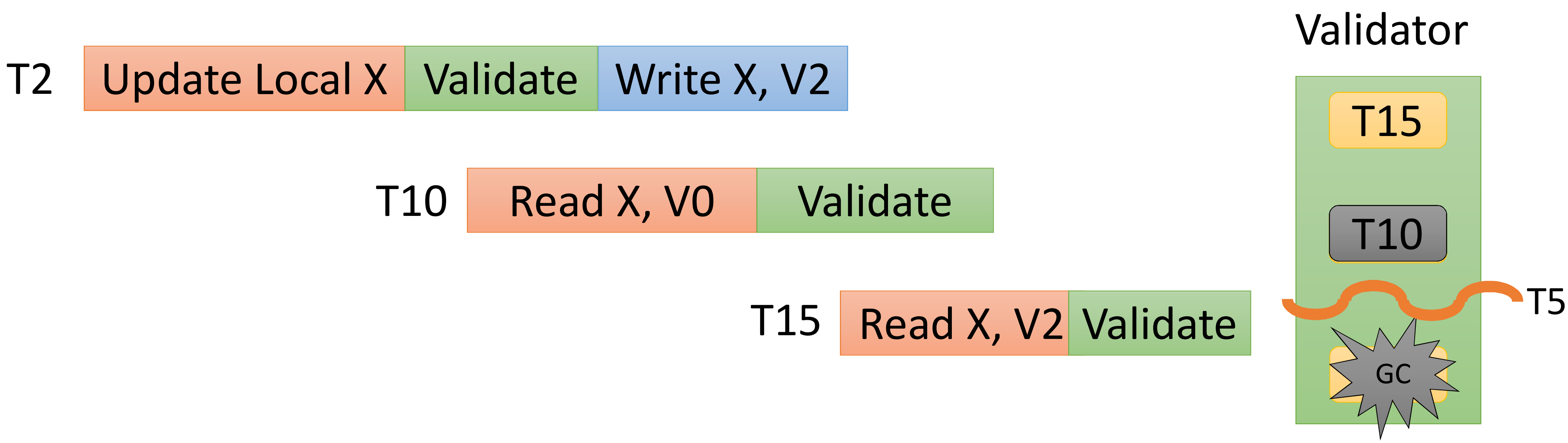
Eliminate Spurious Aborts: Lazy

- Lower the spurious abort rate by reducing the expiration time
- Risk of aggressive garbage collection: abort due to insufficient information



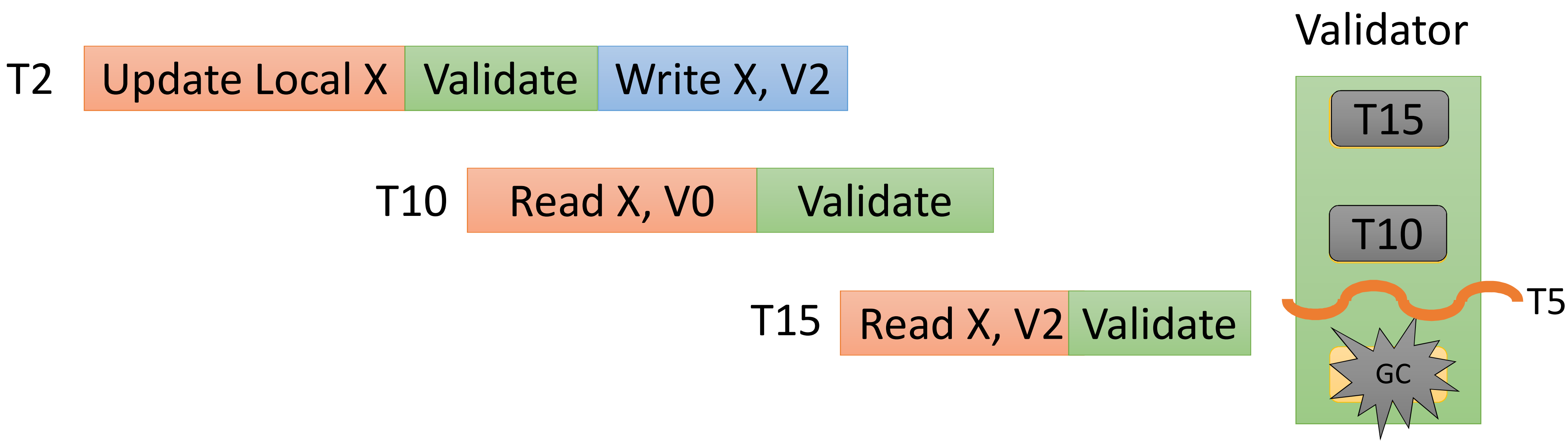
Eliminate Spurious Aborts: Lazy

- Lower the spurious abort rate by reducing the expiration time
- Risk of aggressive garbage collection: abort due to insufficient information



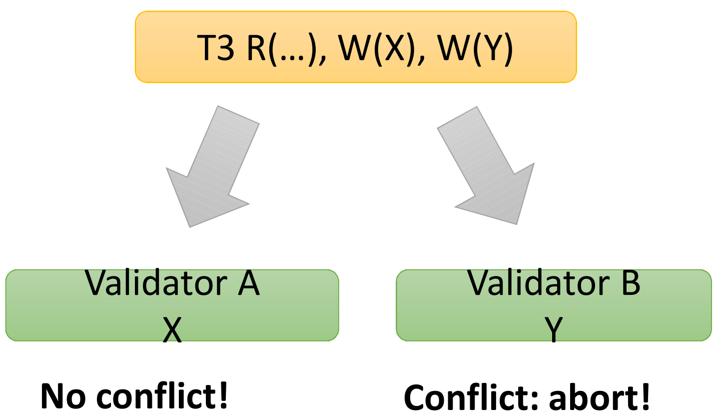
Eliminate Spurious Aborts: Lazy

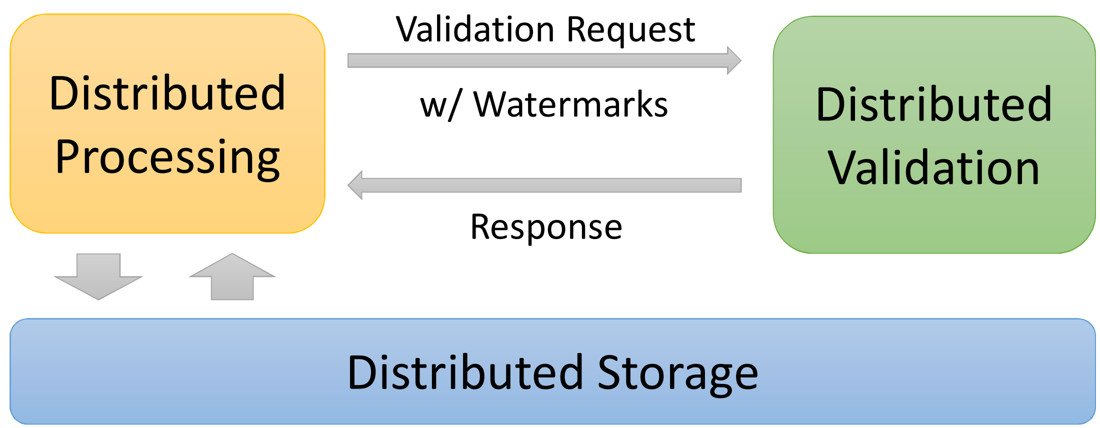
- Lower the spurious abort rate by reducing the expiration time
- Risk of aggressive garbage collection: abort due to insufficient information



Eliminate Spurious Aborts: Reactive

- Garbage collection: Aborts due to insufficient info vs. spurious aborts
- Approach: Asynchronously propagate information by *watermarks*

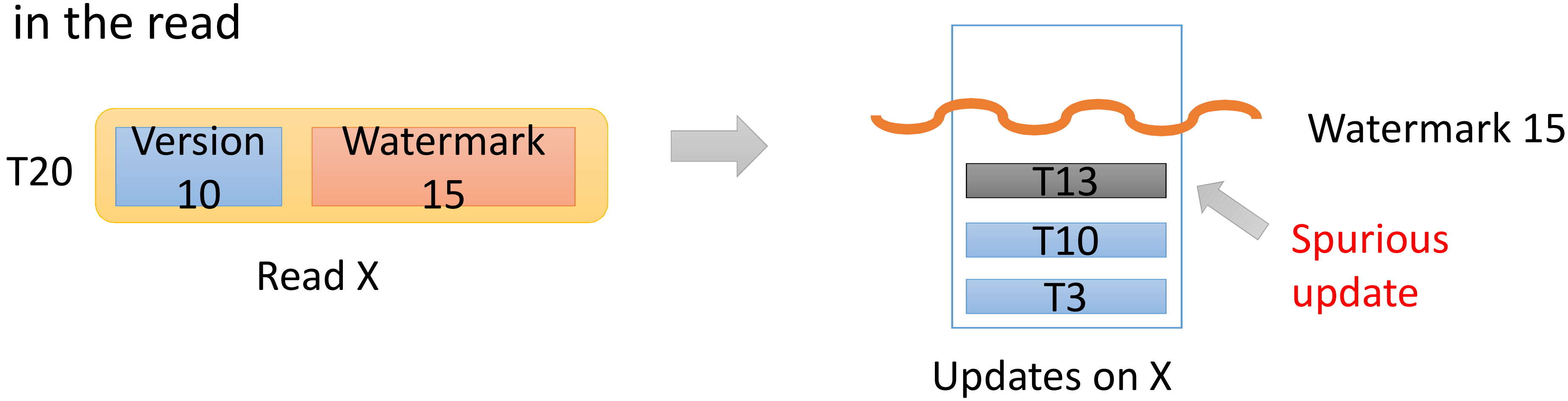




PB26

Validation with Watermarks

- Each read on a record has a watermark
- The watermark is of the same type as the timestamp
- Guarantee: all updates to the record made by transactions with timestamp less than or equal than the watermark have been reflected in the read



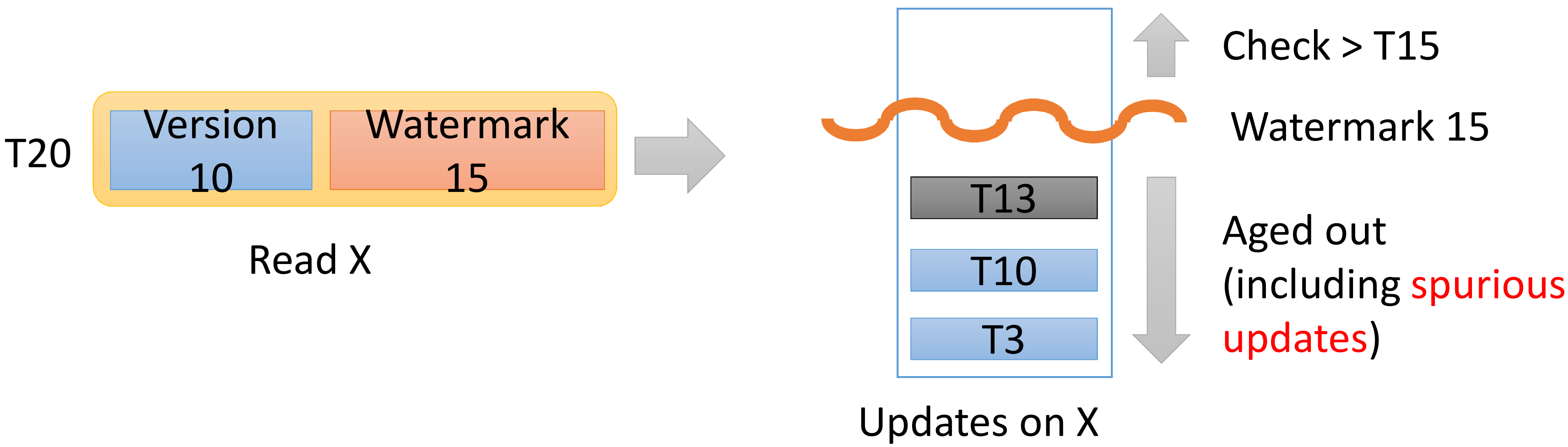
Slide 37

PB26 Rude question: Is timestamp assignment really orthogonal to your system? It could be a bottleneck (if centralized) and could contribute to the abort rate (if transactions arrive late at validators). How will you answer this question?

Phil Bernstein, 2/16/2016

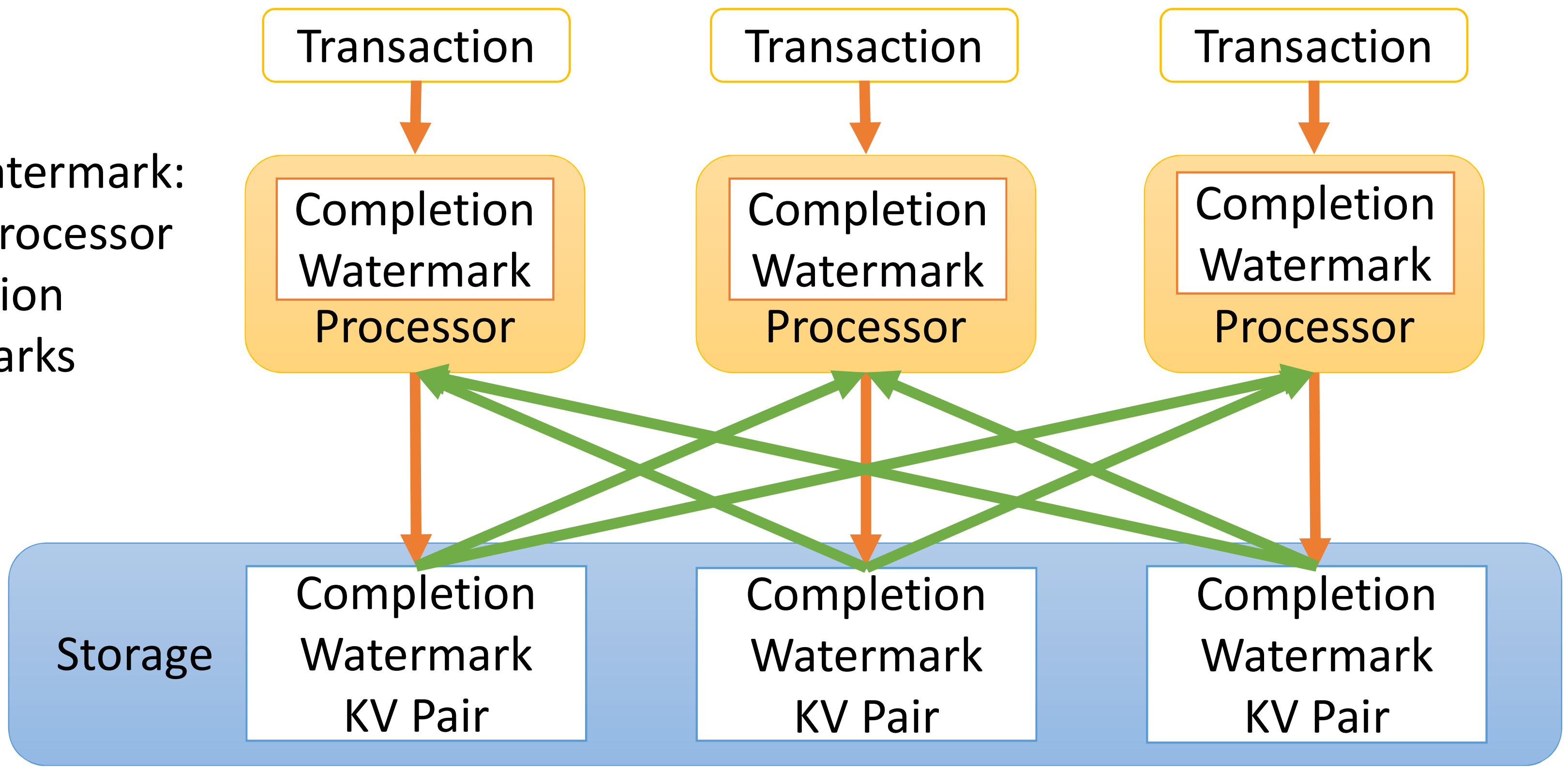
Watermarks Reduce Spurious Aborts

- Spurious updates age out when watermark advances beyond the updating transactions
- Lazy and flexible truncation of history

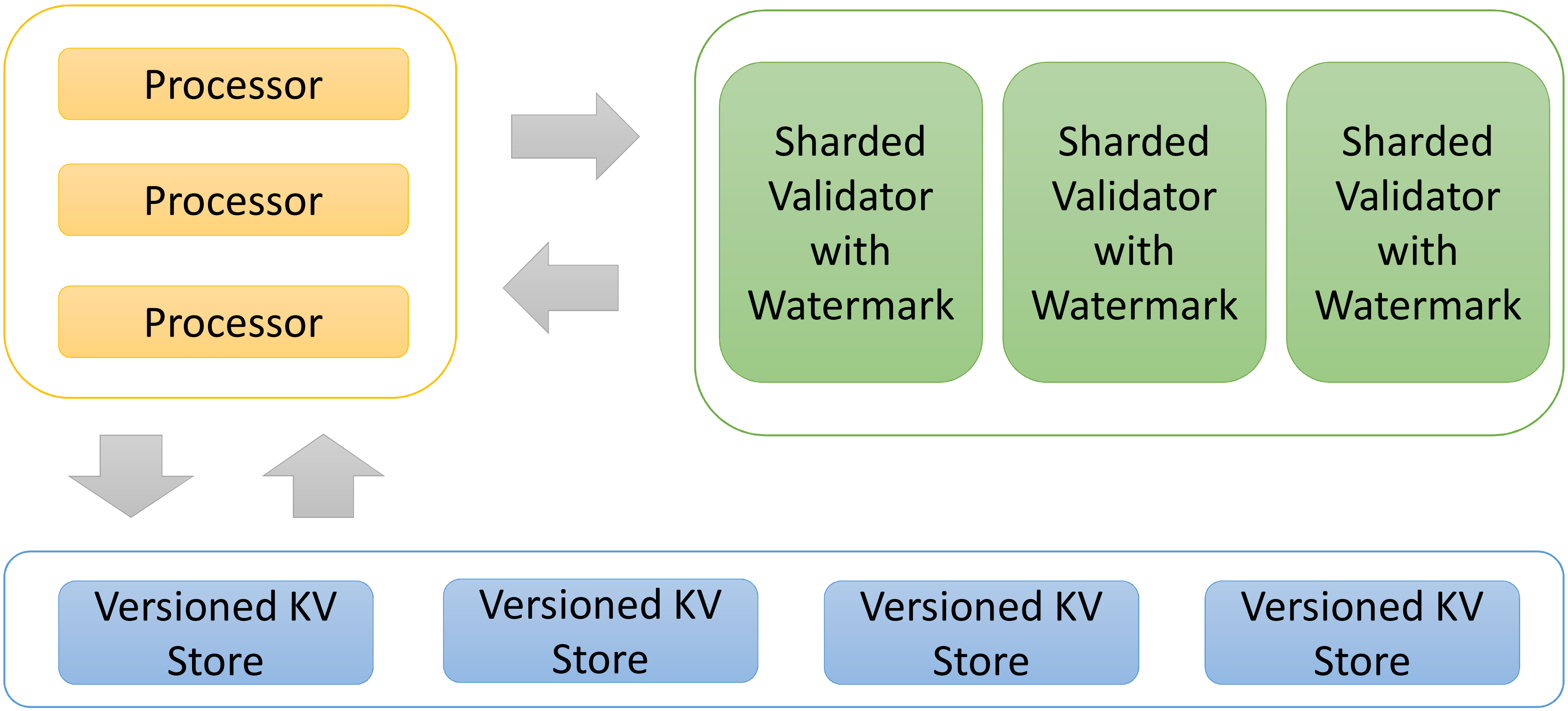


Implementing Watermarks

Read Watermark:
Min of Processor
Completion
Watermarks



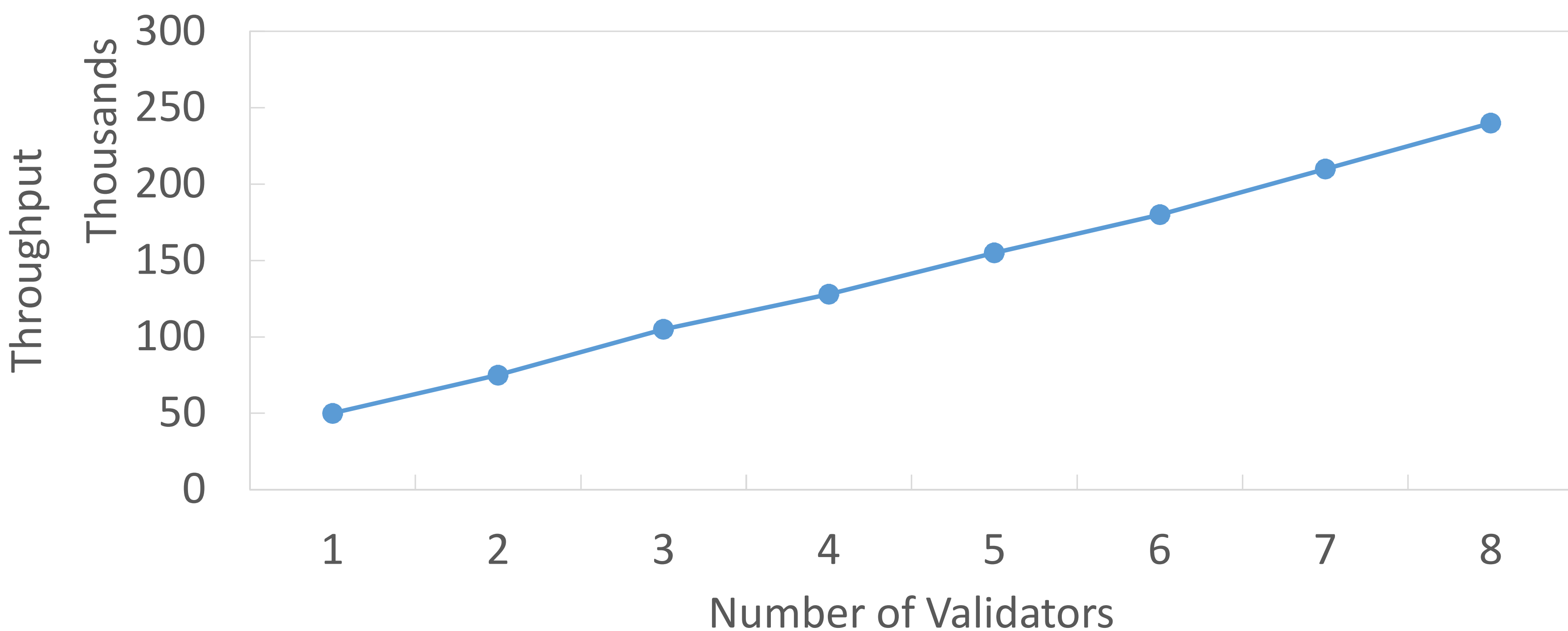
Architecture



Experiment on TPC-C

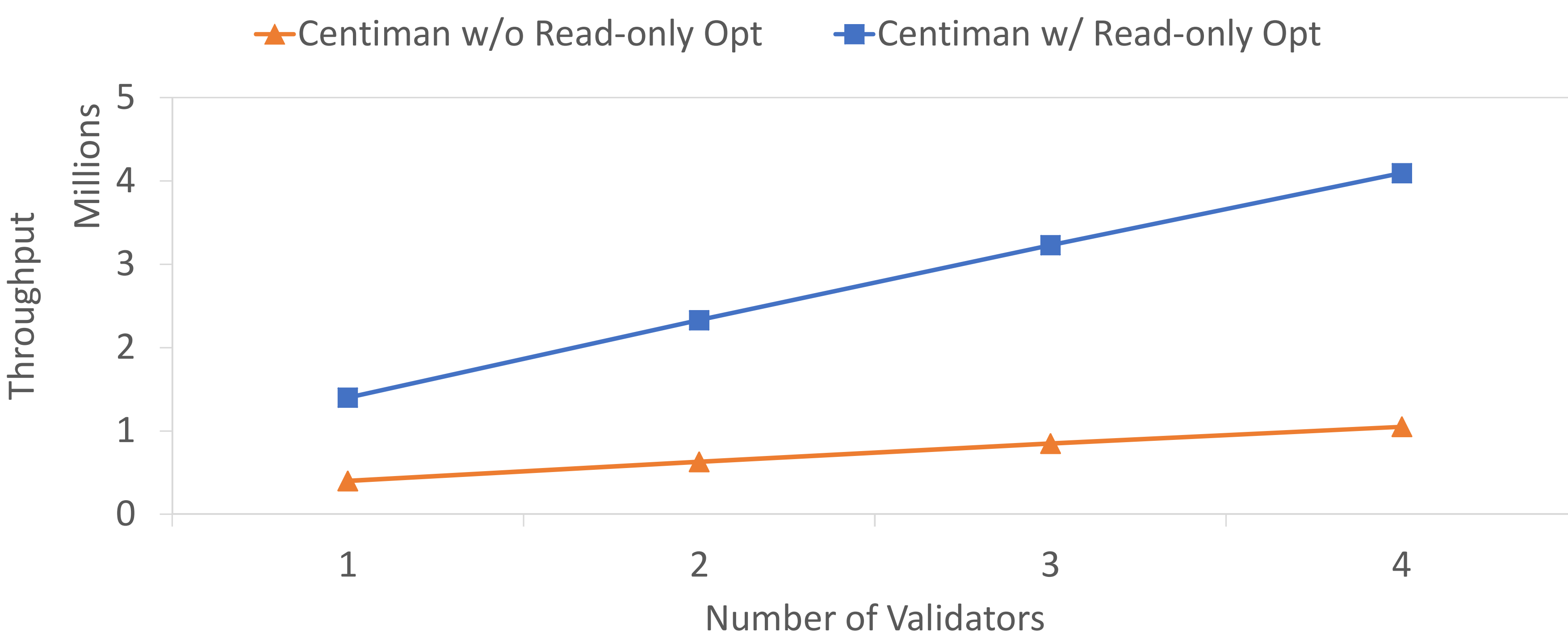
- TPC-C variant
  - Updating transaction only: 50% NewOrder and 50% Payment
- Deployment
  - 50 processor and 50 storage nodes
  - 500 warehouses. Data is randomly shuffled to storage instances
  - 200 concurrent transactions at max per processor

Experimental Results on TPC-C



SD17
SD18
PB58

Experiment Result on TATP

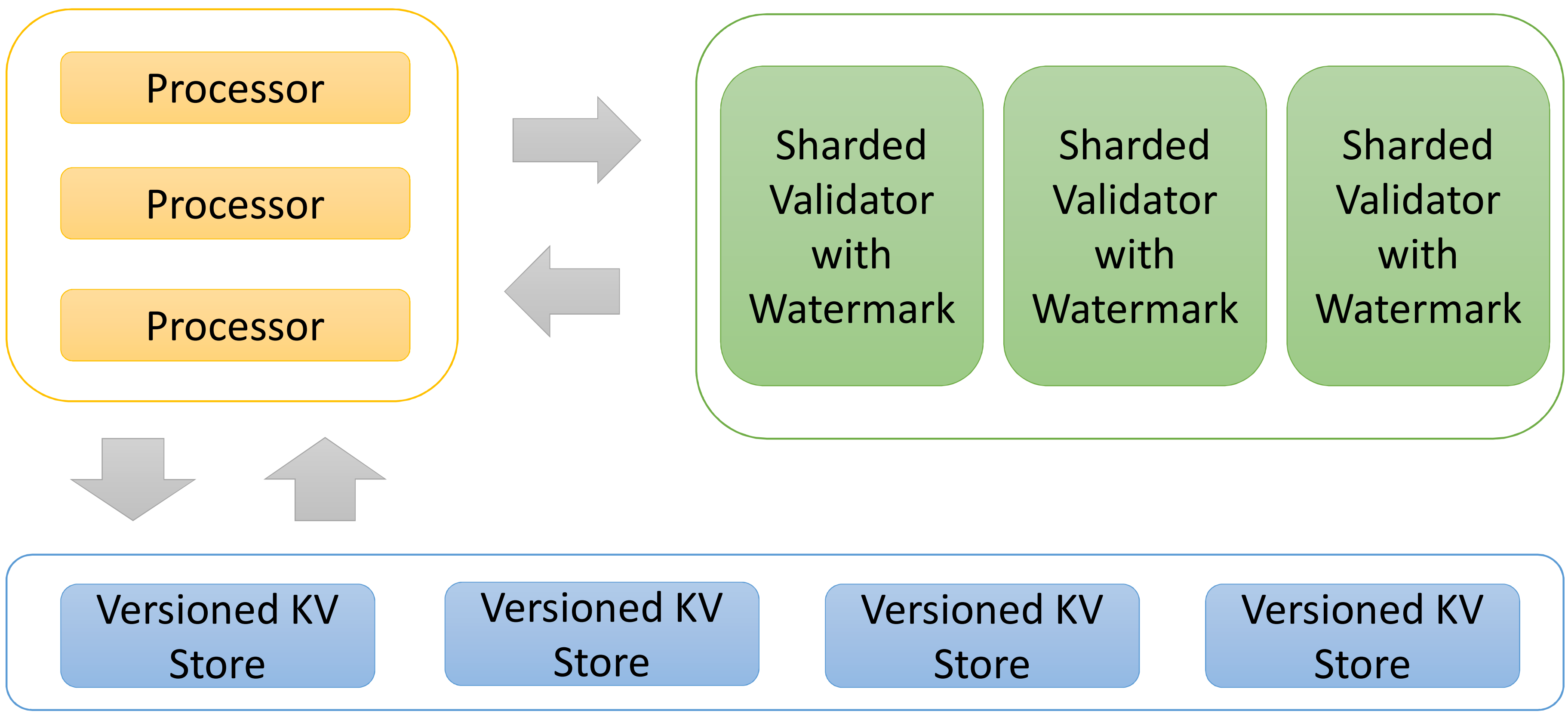


SD17 Rude question: You seem to choose workloads that partition well - for instance TPC-C and TATP. What about performance for workload that don't partition well, for example take the Uniform workload in YCSB-like setting we used for Hyder experiments, or for TPC-E? For TPC-C, were you running the 15% new order transactions that went to a different warehouse? What is the fraction of distributed transactions in your experiments?
Sudipto Das, 2/16/2016

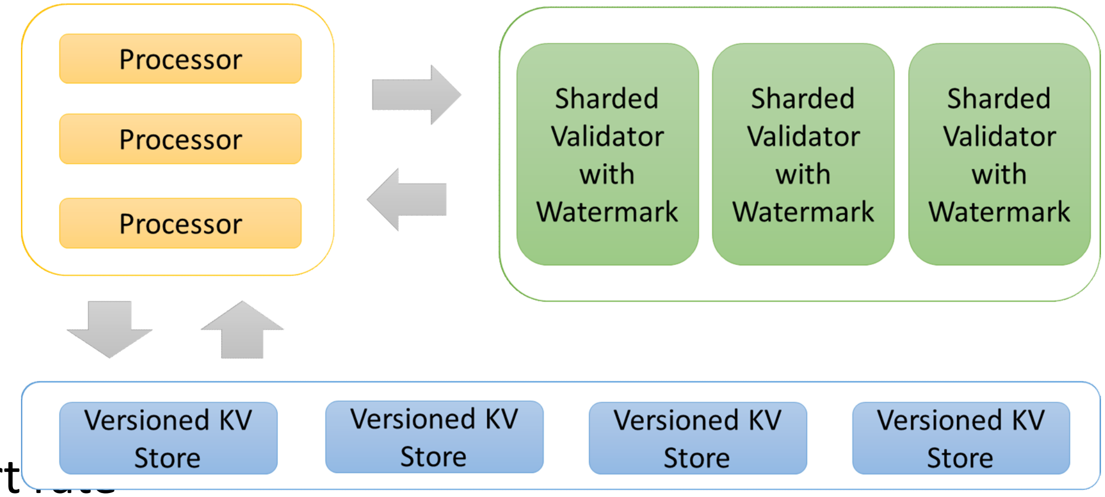
SD18 In general, linear scaling in a distributed transaction processing system will be questioned. Either you didn't stress the system enough for performance to saturate/plateau or thrash, or there is something tricky happening in the experiments. Watch out for those kinds of questions.
Sudipto Das, 2/16/2016

PB58 I agree.
Phil Bernstein, 2/16/2016

Summary: Scaling Out OCC



Extensions



- Optimizations for read-only transactions
- Local re-ordering of transactions to further reduce abort rate
- Use of IEEE Precision Time Protocol (PTP) [1]
  - Synchronize server clocks with < 1 µs precision
  - Permits ordering based on local timestamps
  - Reduces spurious aborts by further ~40%
- Native integration with software-defined flash storage layer

[1] Pulkit A. Misra, Jeffrey S. Chase, Johannes Gehrke, Alvin R. Lebeck: Enabling Lightweight Transactions with Precision Time. ASPLOS 2017: 779-794

What About Distribution?

Distribution

- Who is the most popular wizard in the world of wizards?
  - Transaction 1: Cast one vote for Harry
  - Transaction 2: Cast one vote for Voldemort
  - Transaction 3: Who is leading?

- State replicated across two data centers

West Coast

East Coast

Images: http://www.edgeconnex.com/services/edge-data-centers-proximity-matters/, and http://harrypotter.wikia.com/

Distribution (Contd.)

- T1: Cast one vote for Harry
- T2: Cast one vote for Voldemort
- T3: Who is leading?

West Coast

Voldemort	Harry
100,000	75,000

East Coast

Voldemort	Harry
100,000	75,000

Distribution (Contd.)

- Cast one vote for Harry on the west coast

West Coast

Voldemort	Harry
100,000	75,000

East Coast

Voldemort	Harry
100,000	75,000

Distribution (Contd.)

- Cast one vote for Harry on the west coast

West Coast

Voldemort	Harry
100,000	75,001

East Coast

Voldemort	Harry
100,000	75,001

Distribution (Contd.)

- New state is equivalent to the old state for T1, T2, and T3
  - Equivalent: All transactions will return the same values
- This equivalence will hold for a while

- T1: Cast one vote for Harry
- T2: Cast one vote for Voldemort
- T3: Who is leading?

West Coast

Voldemort	Harry
100,000	75,001

East Coast

Voldemort	Harry
100,000	75,001

Distribution (Contd.)

- New database state:
  - (Voldemort; 100,000), (Harry; 99,999)
- Cast one vote for Harry on the east coast

West Coast

Voldemort	Harry
100,000	99,999

East Coast

Voldemort	Harry
100,000	99,999

Distribution (Contd.)

- Cast one vote for Harry on the east coast
- New state is no longer equivalent to old state for T3 → synchronization is necessary

- T1: Cast one vote for Harry
- T2: Cast one vote for Voldemort
- T3: Who is leading?

West Coast

Voldemort	Harry
100,000	100,000

East Coast

Voldemort	Harry
100,000	100,000

Minimizing Synchronization

- Idea [G1983] : Distribute "equivalence" into slack on both sides
- Avoid synchronization as long as the change is within slack
- Slack is consumed independently at each site without synchronization

West Coast

Voldemort	Harry	Slack
100,000	75,000	12,499

East Coast

Voldemort	Harry	Slack
100,000	75,000	12,500

Minimizing Synchronization (Contd.)

- Many votes

West Coast

Voldemort	Harry	Slack
100,000	75,000	12,499

East Coast

Voldemort	Harry	Slack
100,000	75,000	12,500

Minimizing Synchronization (Contd.)

- Many votes

West Coast

Voldemort	Harry	Slack
100,000	87,499	0

East Coast

Voldemort	Harry	Slack
100,000	75,500	12,000

Minimizing Synchronization (Contd.)

- Many votes
- Synchronize to redistribute slack

West Coast

Voldemort	Harry	Slack
100,000	87,499	0

East Coast

Voldemort	Harry	Slack
100,000	75,500	12,000

Minimizing Synchronization (Contd.)

- Slack has been redistributed

West Coast

Voldemort	Harry	Slack
100,000	87,999	9,000

East Coast

Voldemort	Harry	Slack
100,000	87,999	3,000

Minimizing Synchronization (Contd.)

- Idea: Defer propagation of writes when it is safe to do so
- Much related work on protocols for distributing slack for special situations [G1983,…]

- Problems:
  - Need to re-invent new protocol for each new type of transaction
  - Hard to do manually
  - Error-prone to introduce extra code

- Idea: Given the transaction code, automatically synthesize the right slack distribution protocol

Homeostasis Protocol

- Step 1: Analyze transactions to identify "flexibility" in transactions automatically
  - Intuition: Identify the coarsest granularity at which data must be consistent for correctness → "consistency equivalence classes"

- Step 2: Exploit flexibility in transactions to avoid communication
  - Intuition: Use the equivalence classes to automatically devise protocol that coordinates when necessary

Example

- Two sites

- Two transactions: T1 submitted at Site 1, T2 submitted at Site 2

$$
\begin{aligned}
T_1 ::= \{ & \hat{x} := \texttt{read}(x); \\
& \hat{y} := \texttt{read}(y); \\
& \texttt{if } (\hat{x} + \hat{y} < 10) \texttt{ then} \\
& \quad \texttt{write}(x = \hat{x} + 1) \\
& \texttt{else} \\
& \quad \texttt{write}(x = \hat{x} - 1) \ \}
\end{aligned}
$$

$$
\begin{aligned}
T_2 ::= \{ & \hat{x} := \texttt{read}(x); \\
& \hat{y} := \texttt{read}(y); \\
& \texttt{if } (\hat{x} + \hat{y} < 20) \texttt{ then} \\
& \quad \texttt{write}(y = \hat{y} + 1) \\
& \texttt{else} \\
& \quad \texttt{write}(y = \hat{y} - 1) \ \}
\end{aligned}
$$

Example (Contd.)

- "Tiny" language L
- No loops, but expressive enough to encode all five TPC-C transactions

$$
\begin{aligned}
T_1 ::= \{&\hat{x} := \texttt{read}(x);\\
&\hat{y} := \texttt{read}(y);\\
&\texttt{if } (\hat{x} + \hat{y} < 10) \texttt{ then}\\
&\quad \texttt{write}(x = \hat{x} + 1)\\
&\texttt{else}\\
&\quad \texttt{write}(x = \hat{x} - 1)\ \}
\end{aligned}
$$

$$
\begin{aligned}
T_2 ::= \{&\hat{x} := \texttt{read}(x);\\
&\hat{y} := \texttt{read}(y);\\
&\texttt{if } (\hat{x} + \hat{y} < 20) \texttt{ then}\\
&\quad \texttt{write}(y = \hat{y} + 1)\\
&\texttt{else}\\
&\quad \texttt{write}(y = \hat{y} - 1)\ \}
\end{aligned}
$$

Symbolic Tables

- Analysis computes a symbolic table
  - Mapping from predicates on database to partially evaluated transactions
  - Concise representation of relationship between input and output

$$
\begin{aligned}
T_1 ::= \{&\hat{x} := \texttt{read}(x);\\
&\hat{y} := \texttt{read}(y);\\
&\texttt{if } (\hat{x} + \hat{y} < 10) \texttt{ then}\\
&\quad \texttt{write}(x = \hat{x} + 1)\\
&\texttt{else}\\
&\quad \texttt{write}(x = \hat{x} - 1) \ \}
\end{aligned}
$$

$\varphi_{\mathcal{D}}$	ϕ
$x + y < 10$	$\texttt{w}(x = \texttt{r}(x) + 1)$
$x + y \geq 10$	$\texttt{w}(x = \texttt{r}(x) - 1)$

Analysis Rules

$$[\![T, \{\}]\!] \rightarrow [\![c, \{\langle \texttt{true}, \texttt{skip} \rangle\}]\!] \quad \text{where } T = \{c\}$$

$$[\![c_1; c_2, Q]\!] \rightarrow [\![c_1, [\![c_2, Q]\!]]\!]$$

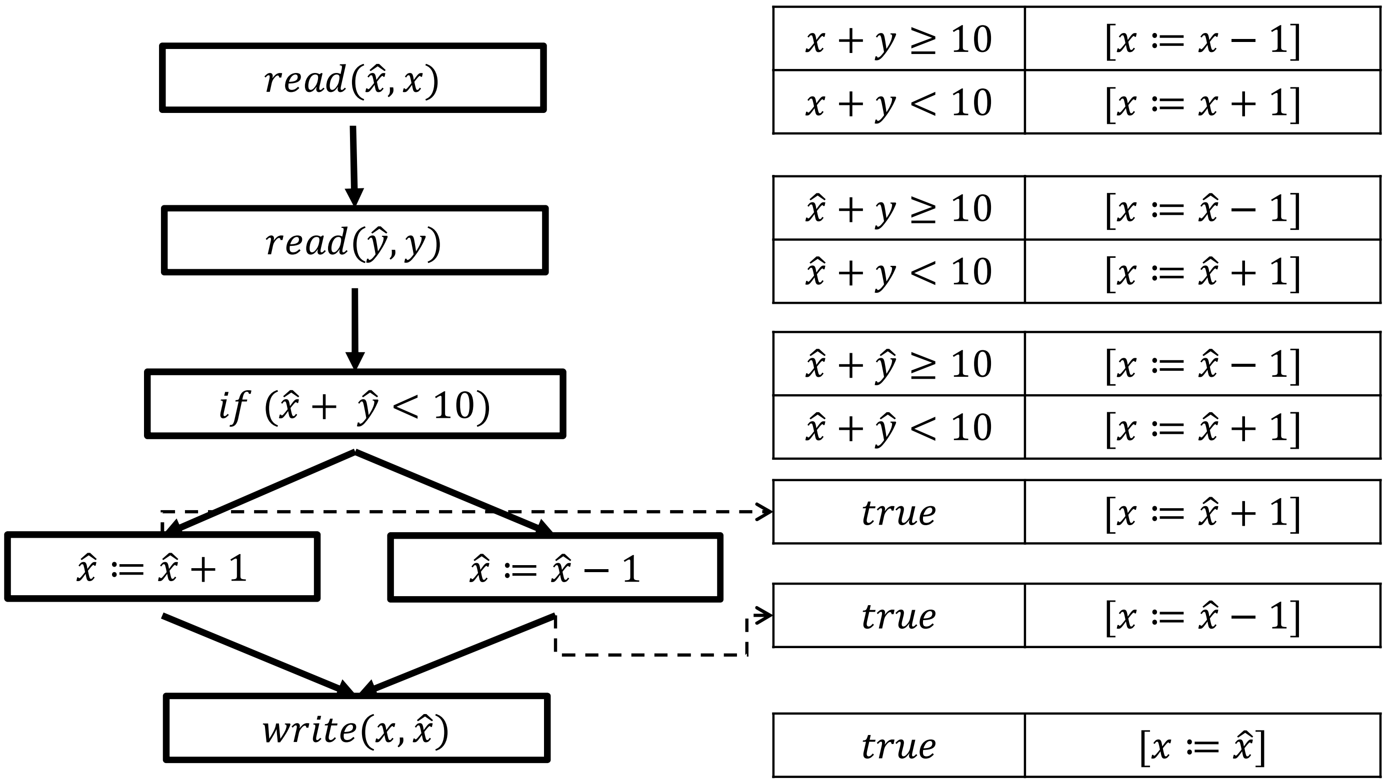
$$\left[\!\!\left[\begin{matrix} \texttt{if}\ b\ \texttt{then}\ c_1 \\ \texttt{else}\ c_2 \end{matrix}, Q \right]\!\!\right] \rightarrow \left\{ \begin{matrix} \{\langle b \wedge \varphi, \phi \rangle \mid \langle \varphi, \phi \rangle \in [\![c_1, Q]\!]\} \cup \\ \{\langle \neg b \wedge \varphi, \phi \rangle \mid \langle \varphi, \phi \rangle \in [\![c_2, Q]\!]\} \end{matrix} \right\}$$

$$[\![(\hat{x} := e), Q]\!] \rightarrow \left\{ \langle \varphi\{\frac{e}{\hat{x}}\}, (\hat{x} := e; \phi) \rangle \mid \langle \varphi, \phi \rangle \in Q \right\}$$

$$[\![\texttt{skip}, Q]\!] \rightarrow Q$$

$$[\![\texttt{write}(x = e), Q]\!] \rightarrow$$
$$\left\{ \langle \varphi\{\frac{e}{x}\}, (\texttt{write}(x = e); \phi) \rangle \mid \langle \varphi, \phi \rangle \in Q \right\}$$

$$[\![\texttt{print}(e), Q]\!] \rightarrow \left\{ \langle \varphi, (\texttt{print}(e); \phi) \rangle \mid \langle \varphi, \phi \rangle \in Q \right\}$$



$read(\hat{x}, x)$
$read(\hat{y}, y)$
$if\ (\hat{x} + \hat{y} < 10)$
$\hat{x} \coloneqq \hat{x} + 1$
$\hat{x} \coloneqq \hat{x} - 1$
$write(x, \hat{x})$
$x + y \geq 10$ | $[x \coloneqq x - 1]$
$x + y < 10$ | $[x \coloneqq x + 1]$
$\hat{x} + y \geq 10$ | $[x \coloneqq \hat{x} - 1]$
$\hat{x} + y < 10$ | $[x \coloneqq \hat{x} + 1]$
$\hat{x} + \hat{y} \geq 10$ | $[x \coloneqq \hat{x} - 1]$
$\hat{x} + \hat{y} < 10$ | $[x \coloneqq \hat{x} + 1]$
$true$ | $[x \coloneqq \hat{x} + 1]$
$true$ | $[x \coloneqq \hat{x} - 1]$
$true$ | $[x \coloneqq \hat{x}]$

Offline Precomputation

- Compute *joint* symbolic table for the complete workload

$$
\begin{aligned}
T_1 ::= \{&\hat{x} := \texttt{read}(x);\\
&\hat{y} := \texttt{read}(y);\\
&\texttt{if } (\hat{x} + \hat{y} < 10) \texttt{ then}\\
&\quad \texttt{write}(x = \hat{x} + 1)\\
&\texttt{else}\\
&\quad \texttt{write}(x = \hat{x} - 1)\ \}
\end{aligned}
$$

$$
\begin{aligned}
T_2 ::= \{&\hat{x} := \texttt{read}(x);\\
&\hat{y} := \texttt{read}(y);\\
&\texttt{if } (\hat{x} + \hat{y} < 20) \texttt{ then}\\
&\quad \texttt{write}(y = \hat{y} + 1)\\
&\texttt{else}\\
&\quad \texttt{write}(y = \hat{y} - 1)\ \}
\end{aligned}
$$

$\varphi_{\mathcal{D}}$	ϕ_1	ϕ_2
$x + y < 10$	$\texttt{w}(x = \texttt{r}(x) + 1)$	$\texttt{w}(y = \texttt{r}(y) + 1)$
$10 \leq x + y < 20$	$\texttt{w}(x = \texttt{r}(x) - 1)$	$\texttt{w}(y = \texttt{r}(y) + 1)$
$x + y \geq 20$	$\texttt{w}(x = \texttt{r}(x) - 1)$	$\texttt{w}(y = \texttt{r}(y) - 1)$

Homeostasis Protocol

- We want to run without synchronization until data changes enough to affect behavior
- We achieve this through treaties
- Global treaty: Invariant on system state
  - Governs how far data can drift before sync
  - Will be computed using symbolic tables
- Local treaties: Local constraints that can be enforced at each site

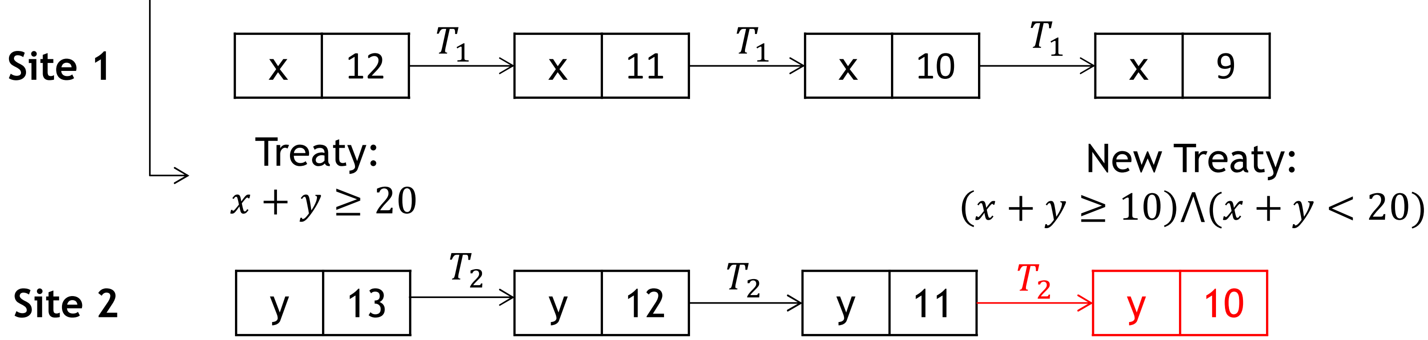
Compute Global Treaty

$\varphi_{\mathcal{D}}$	ϕ_1	ϕ_2
$x + y < 10$	$\mathtt{w}(x = \mathtt{r}(x) + 1)$	$\mathtt{w}(y = \mathtt{r}(y) + 1)$
$10 \leq x + y < 20$	$\mathtt{w}(x = \mathtt{r}(x) - 1)$	$\mathtt{w}(y = \mathtt{r}(y) + 1)$
$x + y \geq 20$	$\mathtt{w}(x = \mathtt{r}(x) - 1)$	$\mathtt{w}(y = \mathtt{r}(y) - 1)$

- The last row of the symbolic table applies
- Global treaty in this case is x+y >= 20

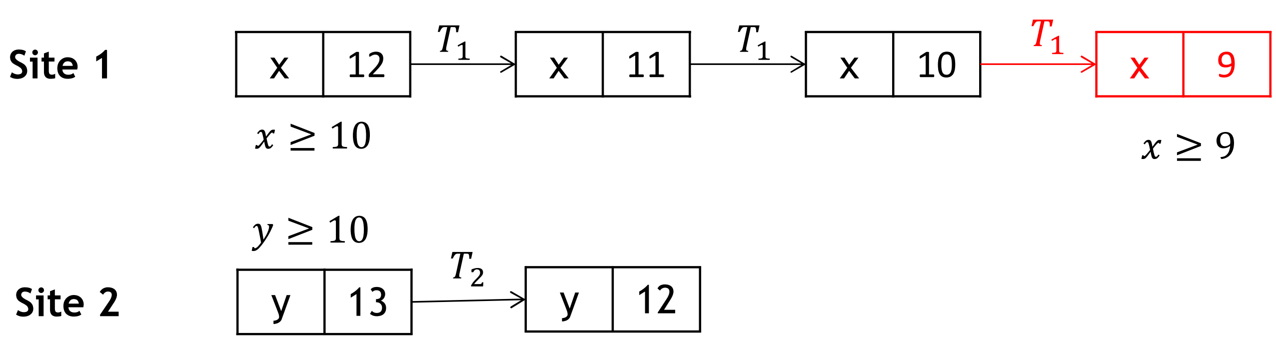
$Q_{\{T_1,T_2\}} :=$

φ_D	ϕ_{T_1}	ϕ_{T_2}
$x + y < 10$	$[x := x + 1]$	$[y := y + 1]$
$(x + y \geq 10) \wedge (x + y < 20)$	$[x := x - 1]$	$[y := y + 1]$
$x + y \geq 20$	$[x := x - 1]$	$[y := y - 1]$



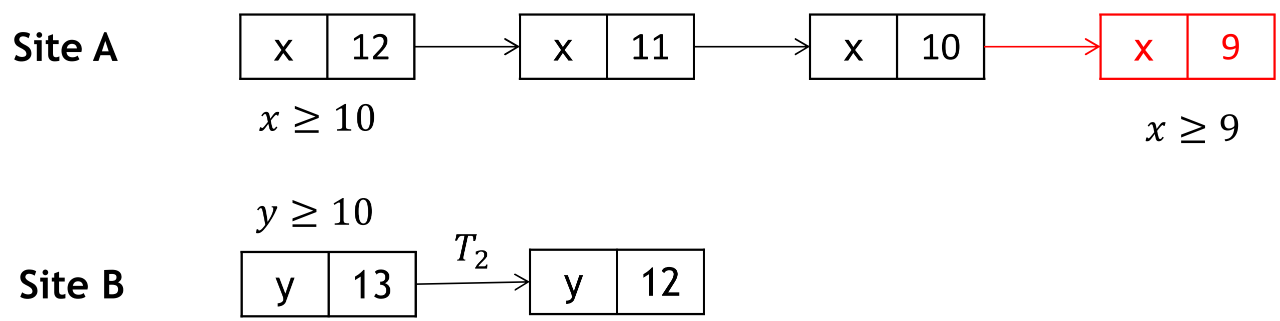
Global and Local Treaties

- Naïve approach: Check global treaty on every write
  - Requires communication on every (update) transaction
- Lazy approach: "Factorize" global treaty into locally enforceable treaties
  - Example: $(x \geq 10) \wedge (y \geq 10) \Rightarrow (x + y \geq 20)$
  - Enforce $(x \geq 10)$ at Site 1 and $(y \geq 10)$ at Site 2



Global and Local Treaties (Contd.)

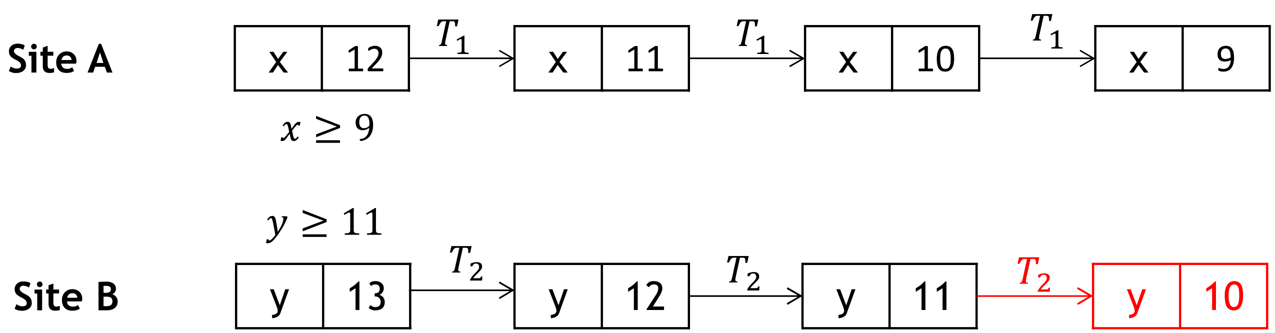
- Multiple correct factorizations of global treaties exist
  - Option 1: $(x \geq 20) \wedge (y \geq 20) \Rightarrow (x + y \geq 20)$
    - Trivially suboptimal
  - Option 2: $(x \geq 10) \wedge (y \geq 10) \Rightarrow (x + y \geq 20)$



Global and Local Treaties (Contd.)

- Multiple correct factorizations of global treaties exist
  - Option 1: $(x \geq 20) \land (y \geq 20) \Rightarrow (x + y \geq 20)$
    - Trivially suboptimal
  - Option 2: $(x \geq 10) \land (y \geq 10) \Rightarrow (x + y \geq 20)$
  - Option 3: $(x \geq 9) \land (y \geq 11) \Rightarrow (x + y \geq 20)$

Optimal for this transaction sequence



Protocol: Summary

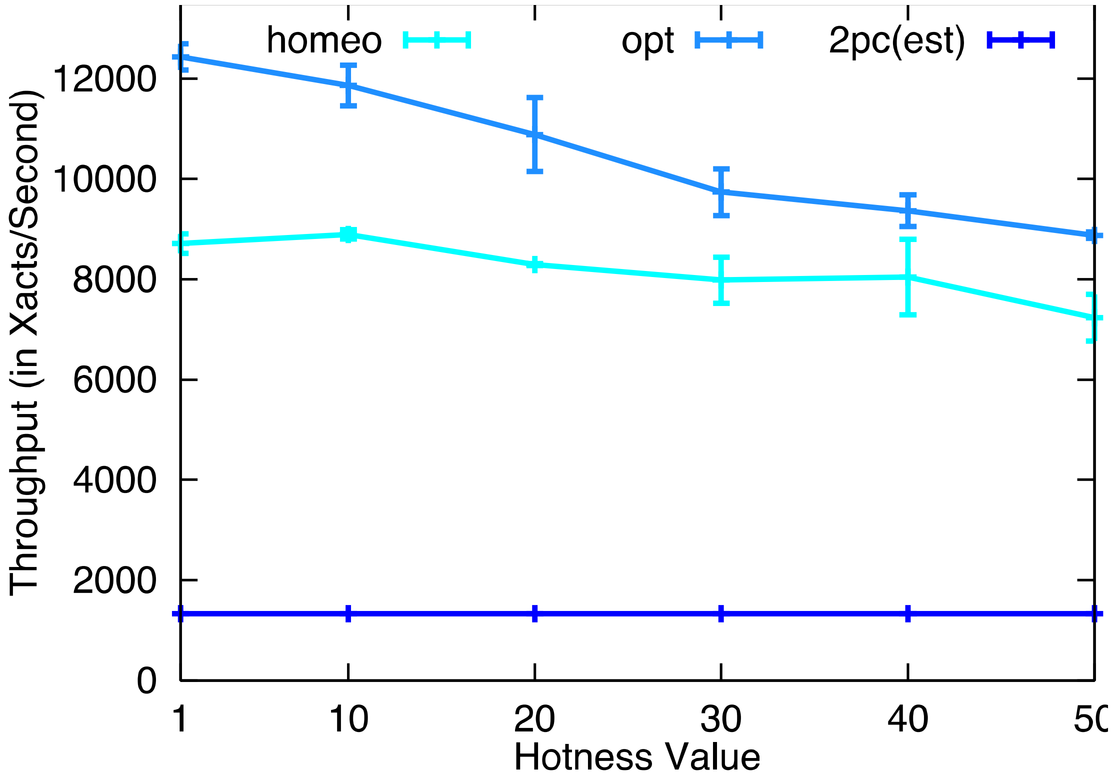
- Compute global treaty using symbolic table
- Factorize into local treaties
- Run disconnected until a local treaty violation occurs
- (Recompute new treaties and continue)

Evaluation

- Replicated system
- Microbenchmark and TPC-C workloads
- Compare against:
  - 2PC (sync at every transaction)
  - local (never sync and lose consistency)
  - opt (hand-coded demarcation protocol)

TPC-C Throughput

- Hotness Value = % of xacts that order the 1% "hot" items



Homeostasis

- Homeostasis protocol reduces need for synchronization without sacrificing consistency
- Fully automated approach based on program analysis
- Provably correct execution

Open Problems

- Expand language for treaties
  - For all of SQL
  - For general programs
  - Combine with replication

- Data layer synthesis
  - To finetune to hardware characteristics
  - To finetune for the workload

Thank you!

johannes@microsoft.com

References

- Bailu Ding, Lucja Kot, Alan J. Demers, Johannes Gehrke: Centiman: Elastic, High-Performance Optimistic Concurrency Control by Watermarking. SoCC 2015: 262-275
- Sudip Roy, Lucja Kot, Gabriel Bender, Bailu Ding, Hossein Hojjat, Christoph Koch, Nate Foster, Johannes Gehrke: The Homeostasis Protocol: Avoiding Transaction Coordination Through Program Analysis. SIGMOD Conference 2015: 1311-1326
- Pulkit A. Misra, Jeffrey S. Chase, Johannes Gehrke, Alvin R. Lebeck: Enabling Lightweight Transactions with Precision Time. ASPLOS 2017: 779-794