DESIGN AND VALIDATION OF CLOUD STORAGE SYSTEMS USING REWRITING LOGIC

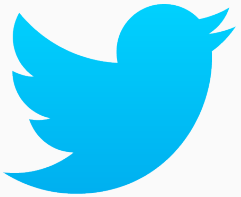
Peter Csaba Ölveczky

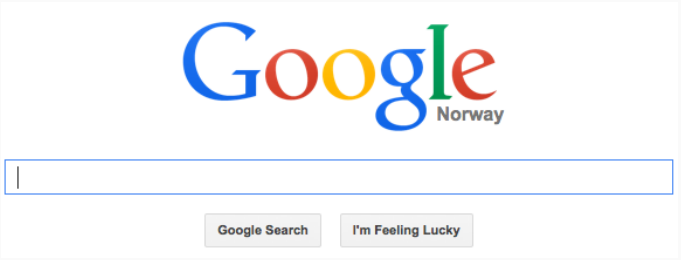
University of Oslo

(based on joint work with Si Liu, José Meseguer, and others)


ALPHA
1 QZ.
2 ABC
3 DEF
4 GHI
5 JKL
6 MNO
7 PRS
8 TUV
9 WXY
0 .SP
#

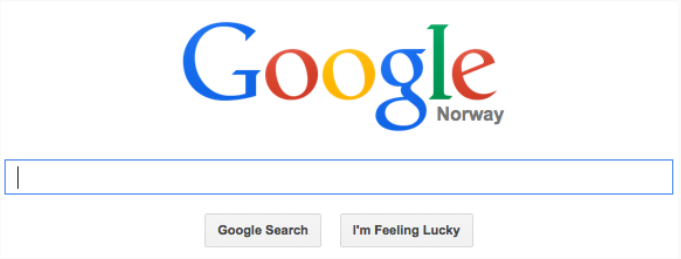
AVAILABILITY





- Data should always be available
  $\longrightarrow$ data replicated

AVAILABILITY



- Data should always be available
  $\longrightarrow$ data replicated
- Large and growing data
  - Facebook (2014): 300 petabytes data; 350M photos uploaded per day

  $\longrightarrow$ data partitioned

REPLICATED SYSTEMS

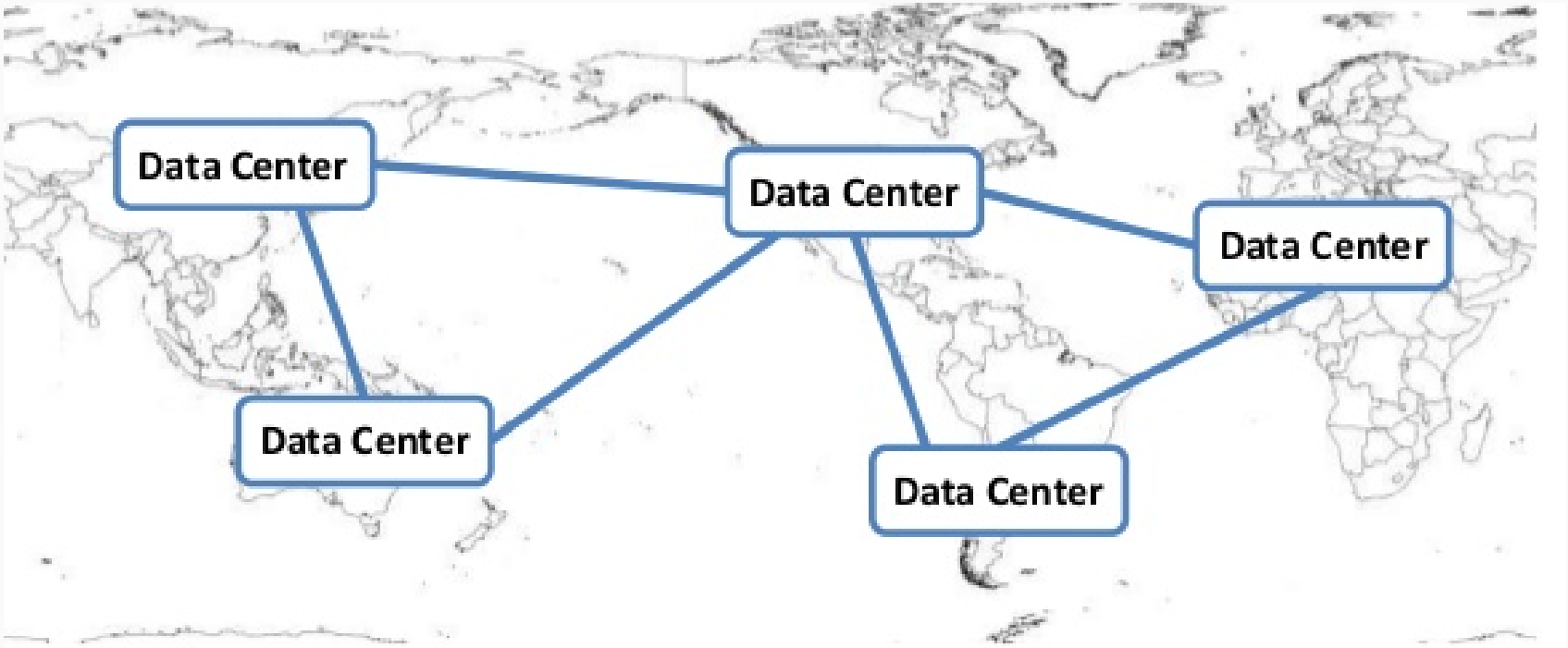
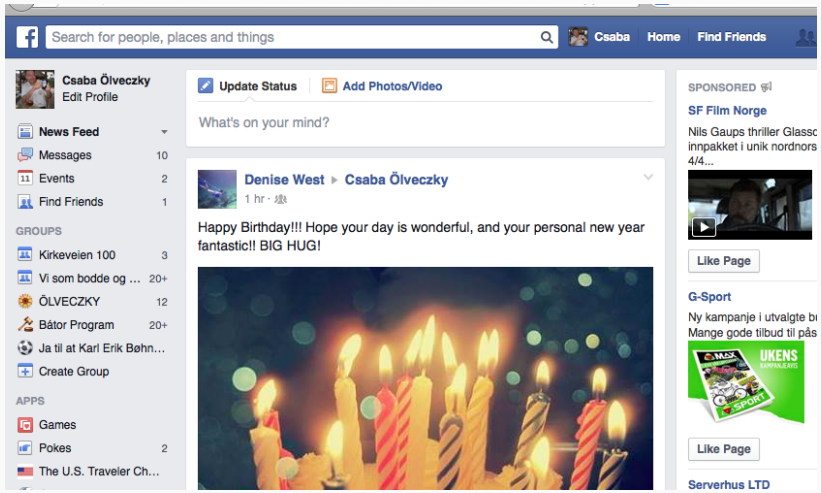


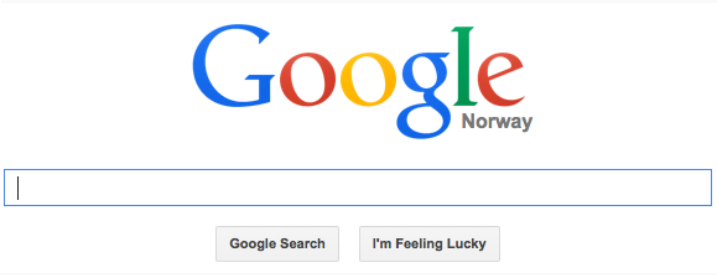
Figure by Jiaqing Du

CONSISTENCY $\longleftrightarrow$ LATENCY

EVENTUAL CONSISTENCY

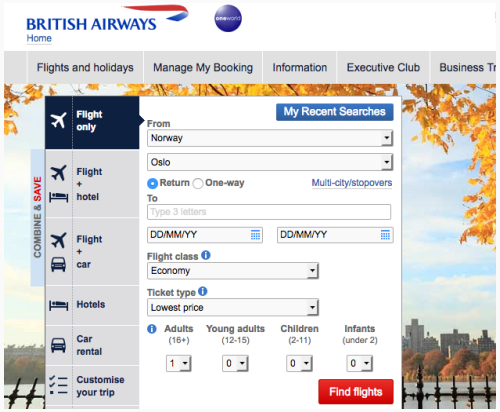
- Weak consistency OK for some applications

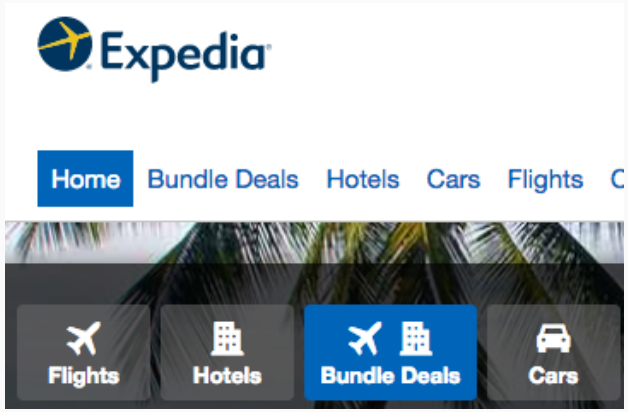




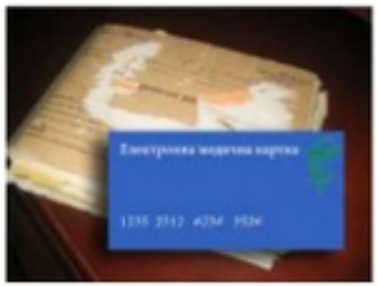
EVENTUAL CONSISTENCY

- Weak consistency OK for some applications
- . . . but not others:





Electronic patient registry

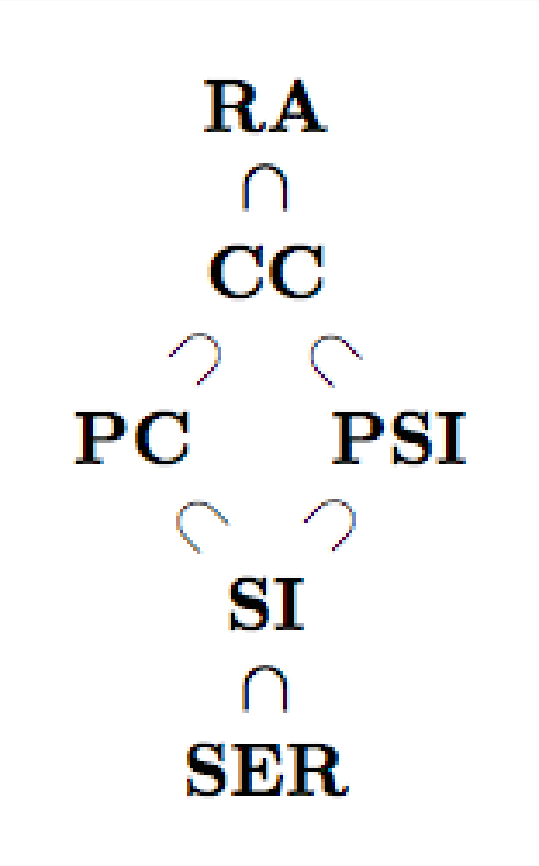


Electronic medical records

SOME CONSISTENCY MODELS

[CERONE, BERNARDI, GOTSMAN; CONCUR'15]

RA	Read Atomic [6]
CC	Causal consistency [19, 12]
PSI	Parallel snapshot isolation [24, 21]
PC	Prefix consistency [13]
SI	Snapshot isolation [8]
SER	Serialisability [20]



SNAPSHOT ISOLATION

Snapshot isolation: All see consistent data

Example

- If Donald see Benny's post before Kim's post
  - then Vlad must also see Benny's post first

SNAPSHOT ISOLATION

Snapshot isolation: All see consistent data

Example

- If Donald see Benny's post before Kim's post
  - then Vlad must also see Benny's post first
  - Benny and Kim must coordinate

SNAPSHOT ISOLATION

Snapshot isolation: All see consistent data

Example

- If Donald see Benny's post before Kim's post
  - then Vlad must also see Benny's post first
  - Benny and Kim must coordinate
- Post cannot commit unless global order ensured

WEAKER CONSISTENCY: PSI/NMSI

Benny and Kim can commit independent posts concurrently

Gary Juffa - Political Agitator for the Indigenous @hunjara · Apr 3

The GENOCIDE IN WEST **PAPUA** CARRIED OUT BY INDONESIAN FORCES AND THAT'S GOING ON WITH THE BLESSING OF THE UN AND WESTERN NATIONS. SINCE 1969 500,000 PLUS DEAD.

> **Mike Cernovich** @Cernovich
> I'm feeling the vibe of doing short films into subjects the media won't cover.
>
> What's a subject you'd like to see get a 20-25 minute short film on?
>
> Show this thread

8 14

Peace for West Papua @WestPapuaPeace · 19h

Free West **Papua** Global Day of Action on april 5th 2019 is an undemocratic action against General Elections in Indonesia.
Please avoid this evil campaign!

CAUSAL CONSISTENCY

Vlad sees reply to Kim's post $\Longrightarrow$ Vlad must also see Kim's post

#FreeAssange! (tweets by campaign) ⏳ Retweeted

Brian Klaas @brianklaas · 54m

They murdered his Dad, dismembered his body, orchestrated a cover-up, lied about the murder, then lied about the cover-up...and now are forcing Khashoggi's son, who is a captive in his own country, to participate in a photo-op shaking hands with the prince who murdered his Dad.

Tom Gara @tomgara

Spare a thought for Khashoggi's son, banned from leaving Saudi Arabia, who had to go and do this today

398 4.6K 5.8K

FORMAL METHODS FOR CLOUD STORAGE SYSTEMS

FORMAL METHODS CHALLENGES

- Large and complex distributed systems
- Complex properties
- Correctness and performance critical

OUR FRAMEWORK: REWRITING LOGIC AND MAUDE

- Modeling: Rewriting logic
  - equational specification defines data types
  - rewrite rules define transitions
  - OO spec: state multiset of messages and objects $< o : Cl \mid att_1 : val_1 , \ldots att_n : val_n >$

OUR FRAMEWORK: REWRITING LOGIC AND MAUDE

- Modeling: Rewriting logic
  - equational specification defines data types
  - rewrite rules define transitions
  - OO spec: state multiset of messages and objects $< o : Cl \mid att_1 : val_1 , \ \ldots \ att_n : val_n >$
- Correctness analysis: Maude
  - simulation, search, LTL model checking

OUR FRAMEWORK: REWRITING LOGIC AND MAUDE

- Modeling: Rewriting logic
  - equational specification defines data types
  - rewrite rules define transitions
  - OO spec: state multiset of messages and objects $< o : Cl \mid att_1 : val_1 , \ldots att_n : val_n >$
- Correctness analysis: Maude
  - simulation, search, LTL model checking
- Performance estimation: PVeStA
  - parallel statistical model checker
  - estimate expected value of expression

MAUDE FOR CLOUD STORAGE SYSTEMS

MODELING

- object-based
- untimed

MODELING

- object-based
- untimed

Example (Transaction Commit Identified)

```
rl [commit-read-only-txn] :
   < RID : Walter-Replica | committed : TRANSES',
                            executing : TRANSES
       < TID : Walter-Txn | operations : nil, writeSet : empty, readSet : RS > >
 =>
   < RID : Walter-Replica | committed : (TRANSES' < TID : Walter-Txn | >),
                            executing : TRANSES > .
```

CORRECTNESS ANALYSIS

- Add "history log" to state
  - records history of execution
  - consistency properties defined on "history"

CORRECTNESS ANALYSIS

- Add "history log" to state
  - records history of execution
  - consistency properties defined on "history"
- Model checking
  - single initial states
  - all initial states with certain parameters

DETAILS: CORRECTNESS ANALYSIS

- Add monitor object

  `< M : Monitor | clock : Nat, log : Log >`

- `clock` counter to order events globally
- Log:

  $id_{txn} \mapsto$ < *proxy*, *issueTime*, *finishTime*, *committed*, *reads*, *writes* >

- Rules transformed by adding & updating Monitor

ADDING HISTORIES II

Example (Transaction Commit Identified)

```
rl [commit-read-only-txn] :


   < RID : Walter-Replica | committed : TRANSES',
                            executing : TRANSES
       < TID : Walter-Txn | operations : nil, writeSet : empty, readSet : RS > >
 =>


   < RID : Walter-Replica | committed : (TRANSES' < TID : Walter-Txn | >),
                            executing : TRANSES > .
```

ADDING HISTORIES II

Example (Transaction Commit Identified)

```
rl [commit-read-only-txn] :
   < M : Monitor | clock : C, log : LOG,
                   (TID |-> < RID, T, VTS, FLAG, READS, WRITES)) >
   < RID : Walter-Replica | committed : TRANSES',
                            executing : TRANSES
       < TID : Walter-Txn | operations : nil, writeSet : empty, readSet : RS > >
 =>
   < M : Monitor | clock : C + 1, log : LOG,
                   (TID |-> < RID, T, insert(RID,C,VTS), true, RS, empty >)
   < RID : Walter-Replica | committed : (TRANSES' < TID : Walter-Txn | >),
                            executing : TRANSES > .
```

DEFINING PROPERTIES

Consistency properties defined on logs in final states

Example

Read Committed

- no aborted data (1st equation)
- no intermediate data (2nd equation)

```
op rc : Log -> Bool .
eq rc(TID1 |-> <O,T,VT,true,(<X,V>,RS),WS>,
      TID2 |-> <O',T',VT',false,RS',(<X,V>,WS')>, LOG) = false .
eq rc(TID1 |-> <O,T,VT,true,(<X,V>,RS),WS>,
      TID2 |-> <O',T',VT',true,RS',(<X,V>,<X,V'>,WS')>,
      LOG)  = false if V < V' .
eq rc(LOG) = true [owise] .
```

ANALYZING PROPERTIES

Search for bad final state:

Example

```
search [1] init < m : Monitor | log : empty, clock : 1 >
            =>!
           C:Configuration   < m : Monitor | log : LOG >
    such that not rc(LOG) .
```

PERFORMANCE ANALYSIS WITH MAUDE

1. (Randomized) simulations

PERFORMANCE ANALYSIS WITH MAUDE

1. (Randomized) simulations
2. Probabilistic analysis (using PVeStA)
    - statistical model checking

DETAILS: STATISTICAL MODEL CHECKING USING PVESTA

- Estimate expected value of expression of a path
- Simulate runs until desired level of confidence reached
  - PVeStA: parallel simulations

DETAILS: STATISTICAL MODEL CHECKING USING PVESTA

- Estimate expected value of expression of a path
- Simulate runs until desired level of confidence reached
  - PVeStA: parallel simulations
- Requires fully probabilistic models
  - rewrite model often nonderministic
    - multiple rules applicable
    - rule applicable to multiple objects

DETAILS: PERFORMANCE ESTIMATION USING SMC

Problem 1

Untimed models unsuitable for performance estimation

Problem 2

Nondeterministic models

DETAILS: PERFORMANCE ESTIMATION USING SMC

Problem 1

Untimed models unsuitable for performance estimation

Problem 2

Nondeterministic models

Solution:

- assume "actor" models: actions triggered by messages

DETAILS: PERFORMANCE ESTIMATION USING SMC

Problem 1

Untimed models unsuitable for performance estimation

Problem 2

Nondeterministic models

Solution:

- assume "actor" models: actions triggered by messages
- message delay sampled probabilistically from dense time interval
  - time!
  - prob(two actions enabled in same state) = 0

TRANSFORMING UNTIMED TO TIMED PROBABILISTIC

Example (on receiving a query)

```
rl [receive-remote-request-prob] :
   msg request(K, TID, VTS) from RID' to RID
   < RID : Walter-Replica | datastore : DS >
 =>
   < RID : Walter-Replica | >
   msg reply(TID, K, choose(VTS, DS[K])) from  RID to RID' .
```

TRANSFORMING UNTIMED TO TIMED PROBABILISTIC

Example (on receiving a query)

```
crl [receive-remote-request-prob] :
    {T, msg request(K, TID, VTS) from RID' to RID}
    < RID : Walter-Replica | datastore : DS >
  =>
    < RID : Walter-Replica | >
    [D, msg reply(TID, K, choose(VTS, DS[K])) from  RID to RID']
    if D := distr(...) .
```

ANALYZING PERFORMANCE METRICS

- Add (timed) monitor
- Define metric on log (in final states)

Example (throughput)

```
op throughput : Configuration -> Float .

eq throughput(REST  < M : Monitor | log : LOG >)
 = committedNumber(LOG) / totalRunTime(LOG) .
```

- Run PVeStA to estimate (expected) value of measure with given confidence levels

MAUDE APPLICATIONS

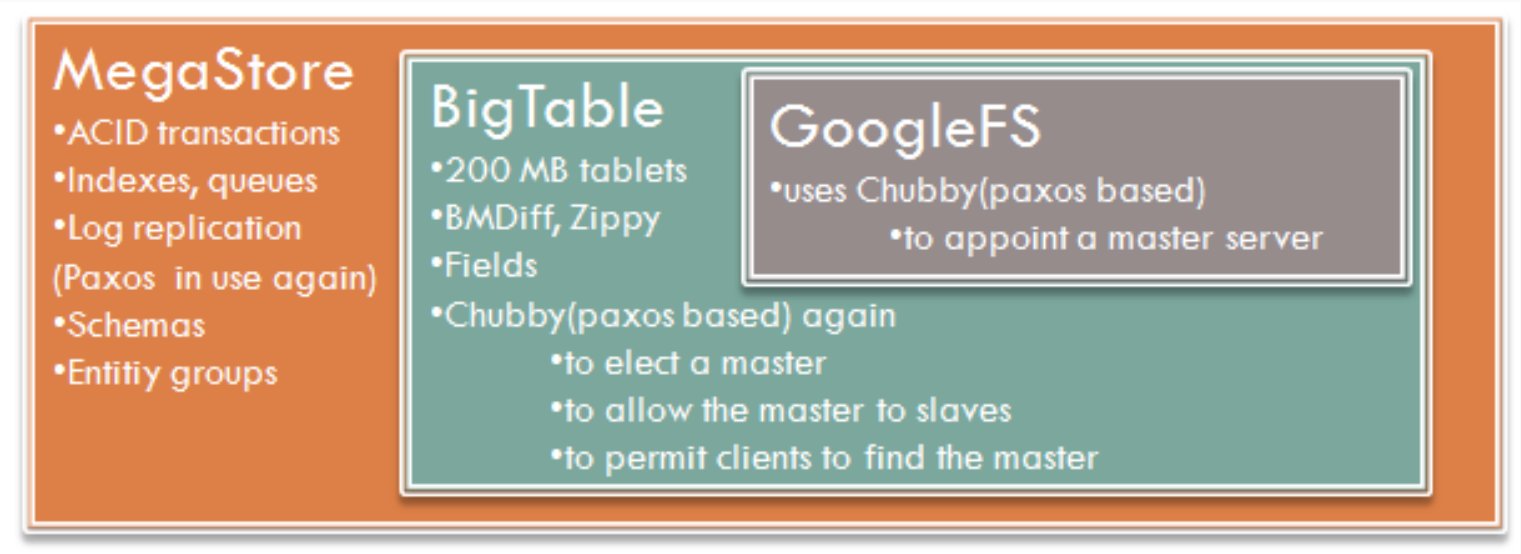
MODELING, ANALYZING, AND EXTENDING MEGASTORE

WITH JON GROV

Megastore:

- Google's wide-area replicated data store
- 3 billion write and 20 billion read transactions daily (2011)

MEGASTORE: KEY IDEAS (I)



(Figure from http://cse708.blogspot.jp/2011/03/megastore-providing-scalable-highly.html)

- Data divided into entity groups
  - Peter's email
  - Books on formal methods
- Consistency for transactions accessing a single entity group

OUR WORK

- [Developed and] formalized [our version of the] Megastore [approach] in Maude
  - first (public) formalization/detailed description of Megastore

OUR WORK

- [Developed and] formalized [our version of the] Megastore [approach] in Maude
  - first (public) formalization/detailed description of Megastore
- 56 rewrite rules (37 for fault tolerance features)

PERFORMANCE ESTIMATION

- Key performance measures:
  - average transaction latency
  - number of committed/aborted transactions

PERFORMANCE ESTIMATION

- Key performance measures:
  - average transaction latency
  - number of committed/aborted transactions
- Randomly generated transactions (2.5 TPS)
- Network delays:

	30%	30%	30%	10%
Madrid $\leftrightarrow$ Paris	10	15	20	50
Madrid $\leftrightarrow$ New York	30	35	40	100
Paris $\leftrightarrow$ New York	30	35	40	100

PERFORMANCE ESTIMATION

- Key performance measures:
  - average transaction latency
  - number of committed/aborted transactions
- Randomly generated transactions (2.5 TPS)
- Network delays:

	30%	30%	30%	10%
Madrid $\leftrightarrow$ Paris	10	15	20	50
Madrid $\leftrightarrow$ New York	30	35	40	100
Paris $\leftrightarrow$ New York	30	35	40	100

- Simulating for 200 seconds:

	Avg. latency (ms)	Commits	Aborts
Madrid	218	109	38
New York	336	129	16
Paris	331	116	21

Megastore-CGC: extending Megastore

MOTIVATION

- Some transactions must access multiple entity groups

MOTIVATION

- Some transactions must access multiple entity groups
- Our work: Megastore + consistency for transactions accessing multiple entity groups

MOTIVATION

- Some transactions must access multiple entity groups
- Our work: Megastore + consistency for transactions accessing multiple entity groups

PERFORMANCE COMPARISON USING REAL-TIME MAUDE

- Simulating for 1000 seconds (no failures)
- Megastore:

	Commits	Aborts	Avg. latency (ms)
Madrid	652	152	126
Paris	704	100	118
New York	640	172	151

- Megastore-CGC:

	Commits	Aborts	Val. aborts	Avg.latency (ms)
Madrid	660	144	0	123
Paris	674	115	15	118
New York	631	171	10	150

CASE STUDY II: APACHE CASSANDRA

Work by Si Liu, Muntasir Raihan Rahman, Stephen Skeirik, Indranil Gupta, José Meseguer, Son Nguyen, Jatin Ganhotra (ICFEM'14, QEST'15)

APACHE CASSANDRA

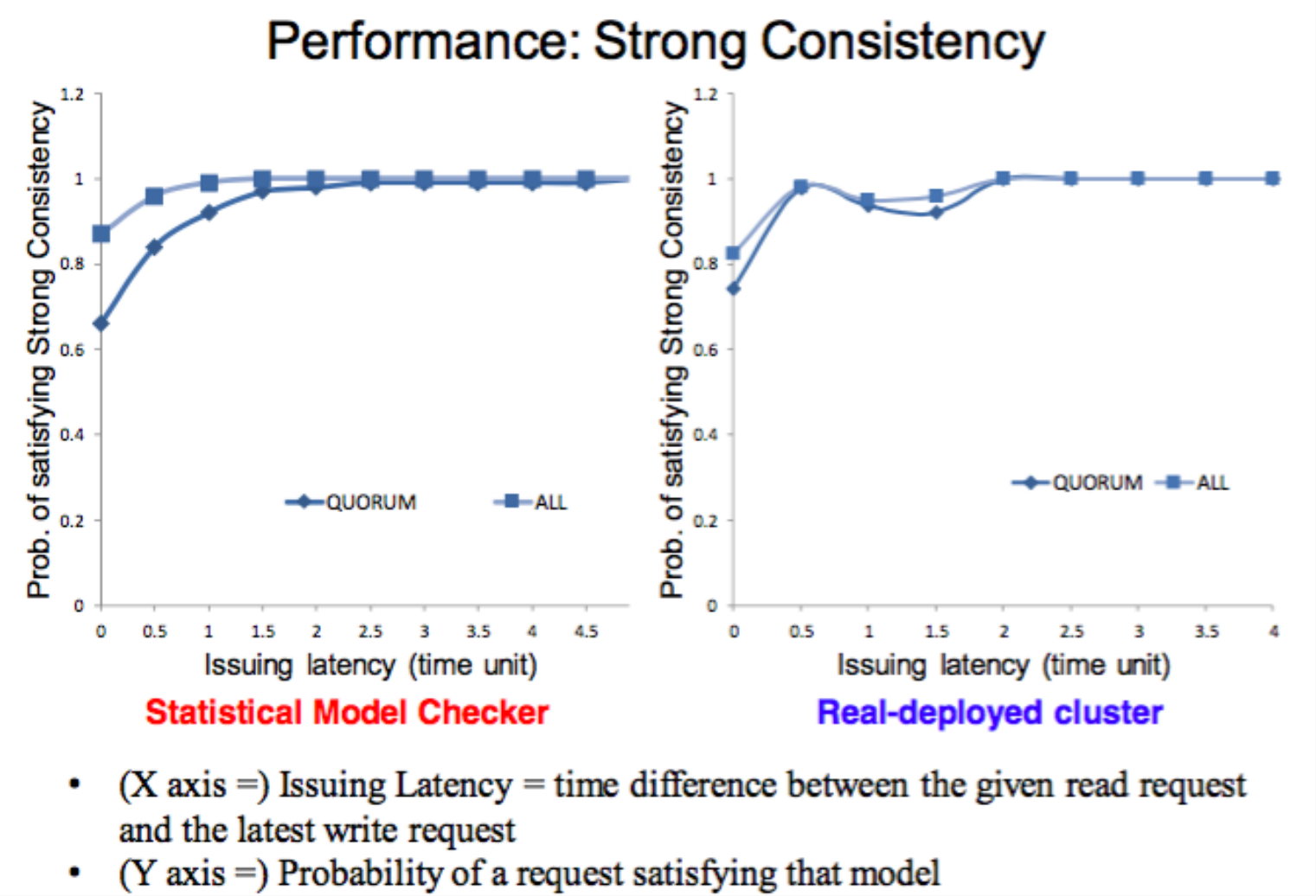
- Key-value data store originally developed at Facebook
- Used by Amadeus, Apple, CERN, IBM, Netflix, Facebook/Instagram, Twitter, ...
- Open source

MOTIVATION

1. Formal model from 345K LOC
    - experiment with different optimizations/variations
2. Analyze consistency properties
3. Performance evaluation:
    - compare PVeStA analyses with real implementations

PERFORMANCE ESTIMATION

Formal model + PVeStA vs. actual implementation



- (X axis =) Issuing Latency = time difference between the given read request and the latest write request
- (Y axis =) Probability of a request satisfying that model

P-STORE [WADT'16]

P-Store [N. Schiper, P. Sutra, and F. Pedone; IEEE SRDS'10]

- Replicated and partitioned data store
- Serializability
- Atomic multicast orders concurrent transactions

ANALYZING P-STORE

Find all reachable final states from init3:

```
Maude> (search init3 =>! C:Configuration .)

Solution 1
C:Configuration --> ...
< c1 : Client | pendingTrans : t1, txns : emptyTransList >
< c2 : Client | pendingTrans : t2, txns : emptyTransList >
< r1 : PStoreReplica | aborted : none,
                       committed : < t1 : Transaction | ... >
< r2 : PStoreReplica | aborted : none,
                       committed : < t2 : Transaction | ... >
...
```

- sites validate transactions
- but client never gets result

ANALYZING P-STORE (CONT.)

```
Solution 5
...
< r1 : PStoreReplica | aborted : none, committed : none,
                       submitted : < t1 : Transaction | ... >, .
< r2 : PStoreReplica | aborted : none,
                       committed : < t2 : Transaction| ... > ...
```

- Host does not validate `t1` even when needed info known

P-STORE SUMMARY

Algorithm $\mathcal{A}_{ge}$
A Genuine Certification Protocol - Code of site s

```
1: Initialization
2:    Votes ← ∅

3: function ApplyUpdates(T)
4:    foreach ∀(k, v) ∈ T.up : k ∈ Items(s) do
5:       let ts be Version(k, s)
6:       w_T[k, v, ts + 1]                          {write to the database}

7: function Certify(T)
8:    return ∀(k, ts) ∈ T.rs s.t. k ∈ Items(s) : ts = Version(k, s)

9: To submit transaction T                                        {Task 1}
10:   A-MCast(T) to Replicas(T)                  {Executing → Submitted}

11: When receive(VOTE, T.id, vote) from s'                        {Task 2}
12:   Votes ← Votes ∪ (T.id, s', vote)

13: When A-Deliver(T)                                             {Task 3}
14:   if T is local then
15:      if Certify(T) then
16:         ApplyUpdates(T)
17:         commit T                                {Submitted → Committed}
18:      else abort T                                {Submitted → Aborted}
19:   else
20:      if ∃(k, -) ∈ T.rs : k ∈ Items(s) then
21:         Votes ← Votes ∪ (T.id, s, Certify(T))
22:         send(VOTE, T.id, Certify(T)) to all s' in WReplicas(T) s.t.
                                                    s' ∉ group(s)
23:      if s ∈ WReplicas(T) then
24:         wait until ∃VQ ∈ VQ(T) :
                        ∀s' ∈ VQ : (T.id, s', -) ∈ Votes
25:         if ∀s' ∈ VQ : (T.id, s', yes) ∈ Votes then
26:            ApplyUpdates(T)
27:            commit T                             {Submitted → Committed}
28:         else abort T                            {Submitted → Aborted}
29:   if s ∈ WReplicas(T) then send T's outcome to Proxy(T)
```

Algorithm $\mathcal{A}_{ge}$
A Genuine Certification Protocol - Code of site s

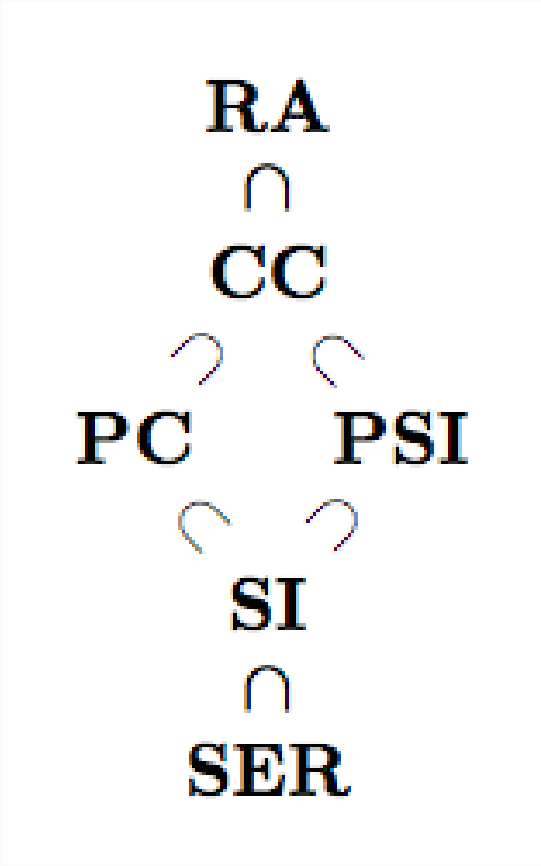
- "P-Store verified"
- 3 significant errors found
- one confusing definition
- key assumption missing

ROLA: UPDATE ATOMIC TRANSACTIONS [FASE'18]

REMEMBER CONSISTENCY MODELS

[CERONE, BERNARDI, GOTSMAN; CONCUR'15]

RA	Read Atomic [6]
CC	Causal consistency [19, 12]
PSI	Parallel snapshot isolation [24, 21]
PC	Prefix consistency [13]
SI	Snapshot isolation [8]
SER	Serialisability [20]



NEW CONSISTENCY MODEL

PSI

read atomic + no lost updates + causal consistency

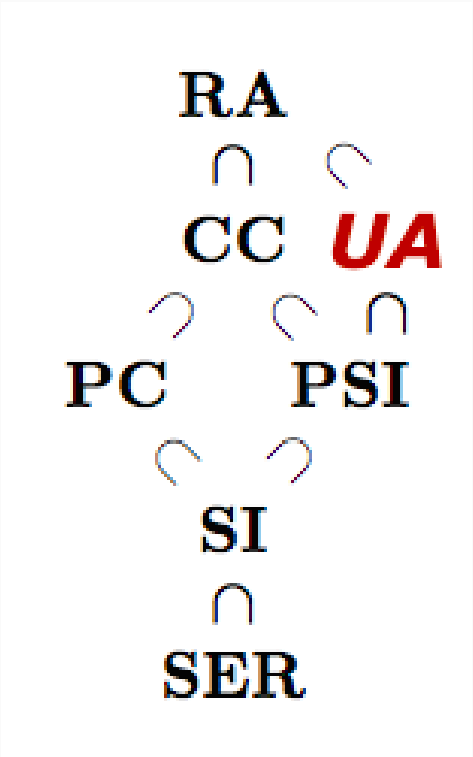
NEW CONSISTENCY MODEL

PSI

read atomic + no lost updates + causal consistency

Update Atomic (UA)

read atomic + no lost updates

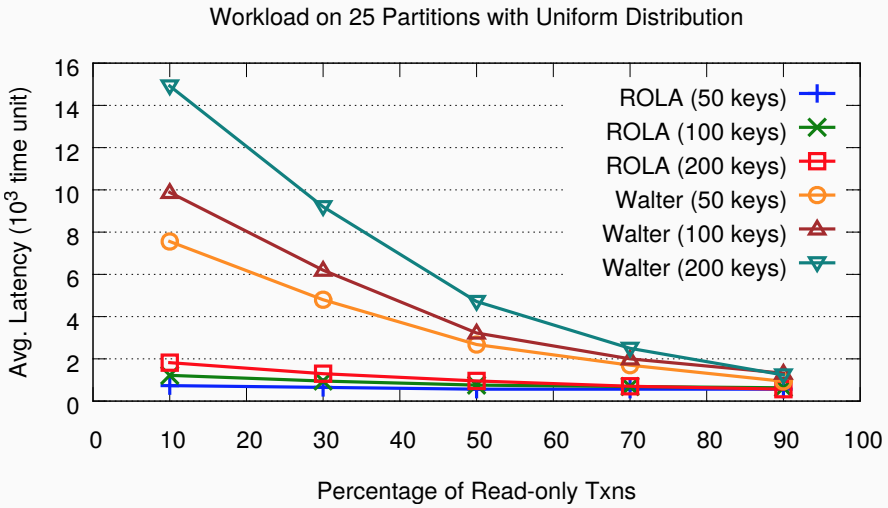


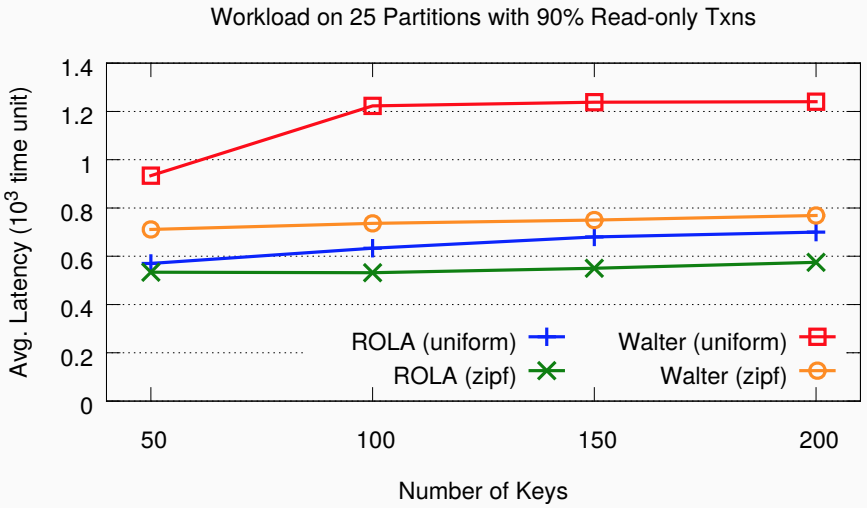
ROLA

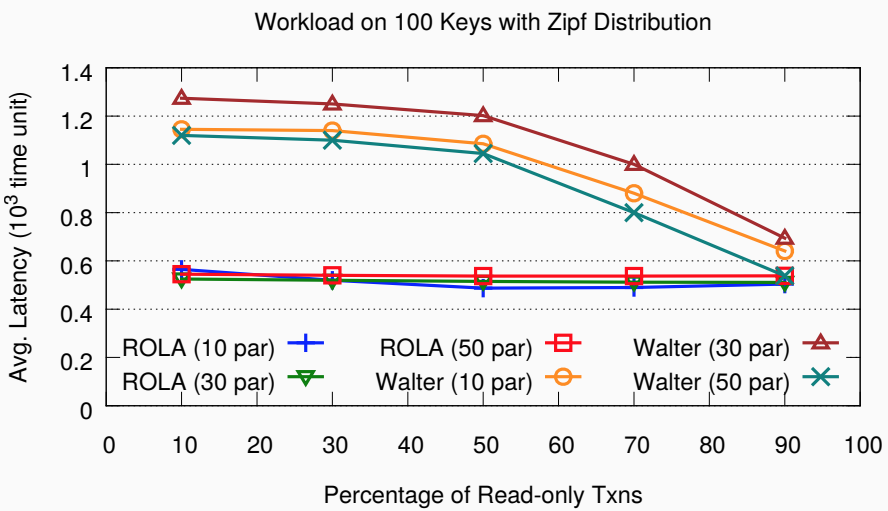
ROLA: new distributed transactions providing UA (= RA + NLU)

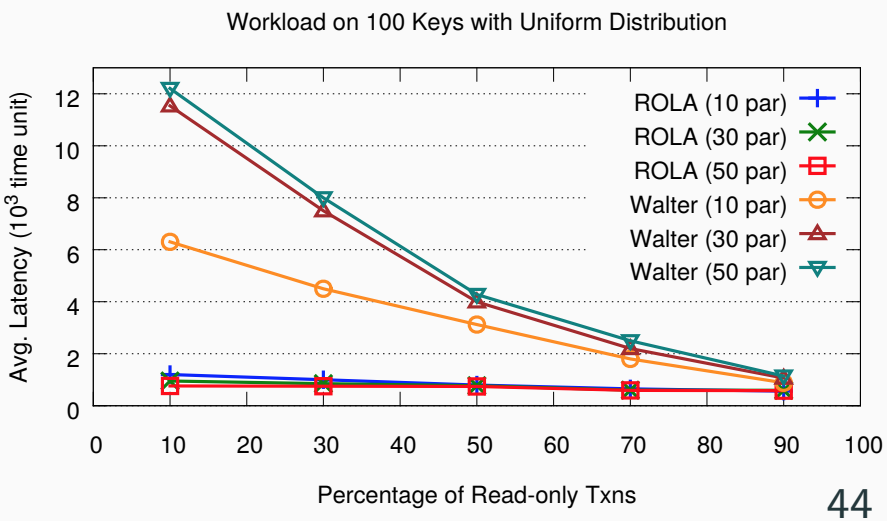
- faster than other designs providing RA+NLU?
  - Walter (PSI)

PERFORMANCE COMPARISON: AVERAGE LATENCY









OTHER DATA STORES

- RAMP (UC Berkeley), Jessy (NMSI), Walter (PSI)
- Extensive performance estimation using PVeStA
  - same trends as original simulations
  - easy to explore many more scenarios

GENERALIZING

GENERALIZING FROM CASE STUDIES

CAT tool

- Maude framework for specifying DTSs
- automatically adds monitoring mechanism
- 9 consistency properties formalized
- generates all initial states up to given bounds
- model checks properties

GENERALIZING FROM CASE STUDIES

CAT tool

- Maude framework for specifying DTSs
- automatically adds monitoring mechanism
- 9 consistency properties formalized
- generates all initial states up to given bounds
- model checks properties

TACAS'19 talk!

CAT ANALYSIS RESULTS

13 DTS Models model checked

- all expected violations found

Maude Model	Consistency Property								
	RC	RA	CS	UA	NMSI	PSI	SI	SER	SSER
RAMP-F	✓	✓	×	×	-	-	×	×	×
RAMP-F+1PW	✓	✓	×	×	-	-	×	×	×
RAMP-F+FC	✓	✓	×	×	-	-	×	×	×
RAMP-F¬2PC	✓	×	×	×	-	-	×	×	×
RAMP-S	✓	✓	×	×	-	-	×	×	×
RAMP-S+1PW	✓	✓	×	×	-	-	×	×	×
RAMP-S¬2PC	✓	×	×	×	-	-	×	×	×
Faster	✓	×	×	×	-	-	×	×	×
ROLA	✓	✓	✓	✓	-	-	×	×	×
Jessy	✓	✓	✓	✓	✓	×	×	×	×
Walter	✓	✓	✓	✓	✓	✓	×	×	×
P-Store	✓	✓	✓	✓	✓	✓	✓	✓	×
NO_WAIT	✓	✓	✓	✓	-	-	✓	✓	×

SUMMARY

SUMMARY

- Developed Maude models of large industrial data stores
  - Google's Megastore (from brief description)
  - Apache Cassandra (from 345K LOC)
  - P-Store and RAMP (academic)

SUMMARY

- Developed Maude models of large industrial data stores
  - Google's Megastore (from brief description)
  - Apache Cassandra (from 345K LOC)
  - P-Store and RAMP (academic)
- Designed own transactional data stores
  - Megastore-CGC
  - variation of Cassandra
  - ROLA

SUMMARY

- Developed Maude models of large industrial data stores
  - Google's Megastore (from brief description)
  - Apache Cassandra (from 345K LOC)
  - P-Store and RAMP (academic)
- Designed own transactional data stores
  - Megastore-CGC
  - variation of Cassandra
  - ROLA
- Automatic model checking analysis of consistency properties

SUMMARY

- Developed Maude models of large industrial data stores
  - Google's Megastore (from brief description)
  - Apache Cassandra (from 345K LOC)
  - P-Store and RAMP (academic)
- Designed own transactional data stores
  - Megastore-CGC
  - variation of Cassandra
  - ROLA
- Automatic model checking analysis of consistency properties
- Maude/PVeStA performance estimation close to real implementations
  - "trends" / "curves" similar to implementations/simulations