An Epistemic Perspective on Consistency of Concurrent Computations

Klaus v. Gleissenthall[1] and Andrey Rybalchenko[1,2]

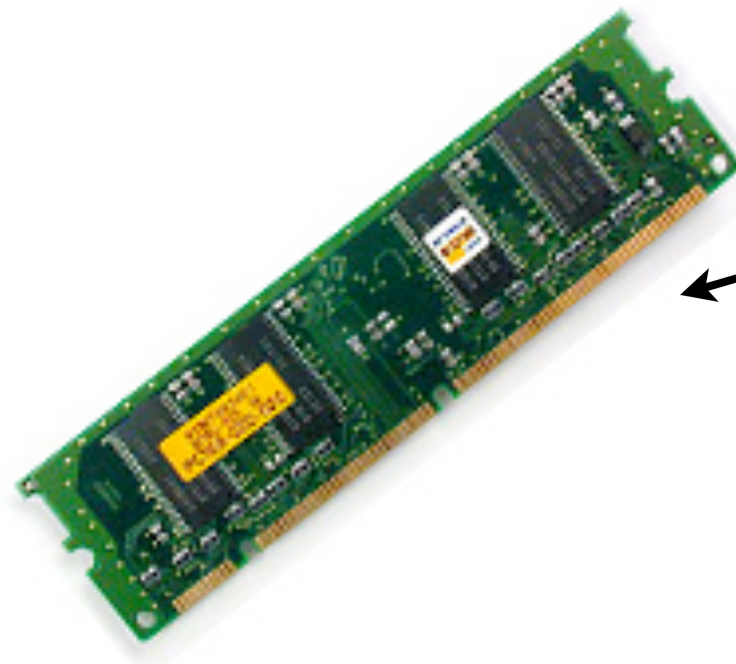
[1] TUM
[2] MSR Cambridge

Consistency Properties

- Whenever read or write shared data at different location
- Need to arrive consensus
- Specify how much inconsistency is tolerated before

Consistency Properties

Memory



Processor
Cache

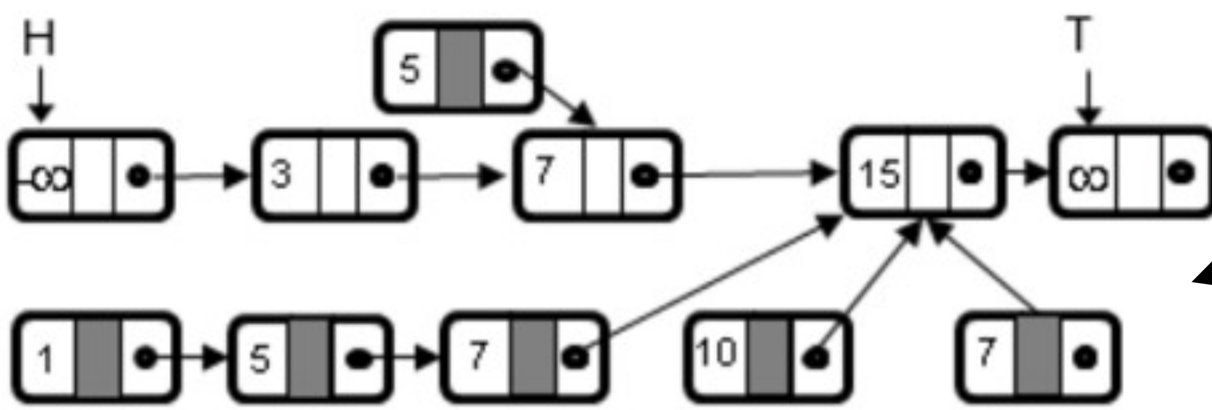
Sequential Consistency/
TSO/
PSO ...

Consistency Properties

Concurrent Data-structures

Non-blocking

Cached Values



Queue/ Set/...

java.util.concurrent

Linearizability

Consistency Properties

Geo-replicated Databases

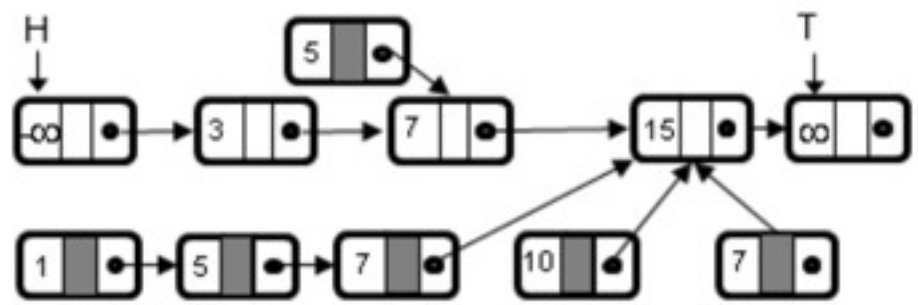
Different
revisions

NoSQL

Eventual Consistency

Consistency Properties

Sequential Consistency/
TSO ...



Linearizability

Eventual Consistency

Different formalisms:

permutation / partial-order / operational

Knowledge Perspective

$\boxed{\text{models}} \xrightarrow{\text{knowledge}} \boxed{\text{logic}}$

Epistemic Logic

Distributed systems

own conference series (TARK)

Distributed Knowledge

trace E

Distributed Knowledge

$$\boxed{E \models D_G(\varphi)}$$

Share everything you know

group G of participants

Halpern and Moses, JACM, 1990

What do we get?

- Very similar form:

For memory:
read last write

$$\boxed{E \models \neg D_G(\neg \mathit{correct})}$$

don't get caught
cheating

Sequential
specification
of shared data-structure

What do we get?

- Compare Conditions:

Sequential consistency: $E \models \neg D_{\mathrm{THREADS}}(\neg \mathit{correct})$

Eventual consistency: $E \models \neg D_{\mathrm{THREADS}}(\neg \mathit{correctEVC})$

Linearizability: $E \models \neg D_{\mathrm{THREADS} \cup \{obs\}}(\neg \mathit{correct})$

What do we get?

- Compare Conditions:

reduction

Sequential consistency: $E \models \neg D_{\text{THREADS}}(\neg \text{correct})$

Eventual consistency: $E \models \neg D_{\text{THREADS}}(\neg \text{correctEVC})$

Linearizability: $E \models \neg D_{\text{THREADS} \cup \{obs\}}(\neg \text{correct})$

additional knowledge

Outline

- Sequential Consistency
- Epistemic Knowledge
- Eventual Consistency
- A Theorem

Sequential-Consistency

Sequential Consistency

arranged by order of return

reason about traces

$$E := (t2, ld(0))\ (t2, ld(1))\ (t1, st(1))$$

is this consistent ?

Sequential Consistency

The result of any execution is the same as if the operations of all the processors were executed in some sequential order, and the operations of each individual processor appear in this sequence in the order specified by its program.

Sequential Consistency

reads return
last write

permutations preserving
program order

E is equivalent to E' and
E' correct wrt. some sequential
specification

Alternative def. from literature

Sequential Consistency

E := (t2, ld(0)) (t2, ld(1)) (t1, st(1)) ✓

reads return last write

E' := (t2, ld(0)) (t1,st(1)) (t2, ld(1))

Knowledge Perspective

t1 "knows" that it stored 1

t2 "knows" that it loaded 0 and then 1

$$E := (t2, ld(0))\ (t2, ld(1))\ (t1, st(1))$$

but: nothing about the other thread

Knowledge Perspective

{t1,t2} know the conjunction of these facts

$$E := (t2, ld(0))\ (t2, ld(1))\ (t1, st(1))$$

i.e., which operations were performed but not their order

Knowledge Perspective

read
last write

$$E \models \neg D_{\{t1,t2\}}(\neg\ \mathrm{correctMem})$$

sequentially
consistent

$$E := (t2, ld(0))\ (t2, ld(1))\ (t1, st(1))$$

Local views

E := (t2, ld(0)) (t2, ld(1)) (t1, st(1))

$\downarrow$ t1

$\downarrow$ t2

projection
onto
t2's view

(t1, st(1))

(t2, ld(0)) (t2, ld(1))

Indistinguishability

$E \sim_t E'$ iff: $E{\downarrow}t = E'{\downarrow}t$

E := (t2, ld(0)) (t2, ld(1)) (t1, st(1))

~t1 ✓ ~t2 ✗

E' := (t2, ld(0)) (t1, st(1))

Group Indistinguishability

$\sim_G := (\cap_{a \in G} \sim_a)$

$E := (t2, ld(0))\ (t2, ld(1))\ (t1, st(1))$

$\sim \{t1, t2\}$

$E' := (t2, ld(0))\ (t1, st(1))$

Group Indistinguishability

$\sim_G := (\cap_{a \in G} \sim_a)$

E := (t2, ld(0)) (t2, ld(1)) (t1, st(1))

$\sim$ {t1,t2} ✓

E' := (t2, ld(0)) (t1,st(1)) (t2, ld(1))

Distributed Knowledge

threads can't tell which trace
they really saw

$$E \models D_G(\varphi) \text{ :iff for all } E' \text{ s.t. } E \sim_G E' \text{: } E' \models \varphi$$

if phi holds for all those traces,
they know phi

Sequential Consistency

$E \models D_G(\varphi)$:iff for all E' s.t. $E \sim_G E'$: $E' \models \varphi$

↓

$E \models \neg D_{\text{THREADS}}(\neg \text{correct})$

:iff exists E' s.t. $E \sim_{\text{Threads}} E'$ and $E' \models \text{correct}$

The threads can't tell E from a correct trace

A good reason to accept it!

Sequential Consistency

E is equivalent to E' and
E' correct wrt. some sequential
specification

$E \models \neg D_{\mathrm{THREADS}}(\neg \mathrm{correct})$
:iff exists E' s.t. $E \sim_{\mathrm{Threads}} E'$ and $E' \models \mathrm{correct}$

Eventual-Consistency

Eventual Consistency

- Geo-replicated database systems (Google/Facebook ...)
- Different location need to maintain consistent view of data
- However must be highly available
- Minimize synchronization, allow updates any time

Original Definition

Existence of two orders

Definition 4 (Eventual Consistency). *We adapt the definition presented in [4] to our notation. A trace $E \in \mathcal{E}^\infty$ is eventually consistent ($evCons(E)$) if and only if there exist a partial order $\prec_v$ (visibility order), and a total order $\prec_a$ (arbitration order) on the events in $set(E)$ such that:*

- $\prec_v \subseteq \prec_a$ *(arbitration extends visibility).*
- $\prec_p \subseteq \prec_v$ *(visibility is compatible with program-order).*
- *for each* $e_q = (t, qu(id, q, r)) \in E$*, we have* $r = apply(\{e \mid e \prec_v e_q\}, \prec_a, s_0)$ *(consistent query results).*
- $\prec_a$ *and* $\prec_v$ *factor over* $\equiv_t$ *(atomic revisions).*
- *if* $(t, com(id)) \notin E$ *and* $(t, \_(id, \_)) \prec_v (t', \_)$ *then* $t = t'$ *(uncommitted updates).*
- *if* $e = (t, com(id)) \in E$ *then there are only finitely many* $e' := (t', com(id'))$ *such that* $e' \in E$ *and* $e \not\prec_v e'$ *(eventual visibility).*

Our Version

Participants don't know it violates

$$E \models \neg D_{\text{THREADS}}(\neg \text{correctEVC})$$

temporal/sequential specification

Eventual Consistency

"so far"

$$\text{correctEVC} := \forall t \forall q \forall r \,(\, \boxminus(\text{query}(t, q, r) \rightarrow \exists L(L \text{ validLog } t \wedge \text{result}(q, L, r)))) \wedge \text{atomicTrans} \wedge \text{alive} \wedge \text{fwd}$$

results are justified by consistent logs

Eventual Consistency

store/load

t knows a

$$L \text{ validLog } t := \forall a (a \text{ in } L \leftrightarrow t\ k_{log}\ a)$$

$$\wedge \text{ consistent } (L)$$

log order matches

concensus

Eventual Consistency

$$(E,i) \models t \; k_{log} \; a \text{ :iff}$$
$$\text{there is } j \leq i : (E@j = (t,a) \text{ or}$$
$$((E,j) \models \mathrm{forward}(t',t,id) \text{ and}$$
$$\text{there is } l < j : (E,l) \models \mathrm{commit}(t',id) \text{ and}$$
$$(E,l) \models t' \; k_{log} \; a))$$

A theorem

Axioms of knowledge

$(T) := \models D_G(\varphi) \rightarrow \varphi$ (Truth axiom)

$(4) := \models D_G(\varphi) \rightarrow D_G(D_G(\varphi))$ (Pos. Introspection)

$(5) := \models \neg D_G(\varphi) \rightarrow D_G(\neg D_G(\varphi))$ (Neg. Introspection)

Knowledge about Consistency

$$\text{seqCons} := \neg D_{\text{THREADS}}(\neg\text{correct})$$

$$\text{lin} := \neg D_{\text{THREADS} \cup \{\text{obs}\}}(\neg\text{correct})$$

Knowledge about Consistency

$$\models (\mathit{seqCons} \leftrightarrow D_{\mathit{Threads}}(\mathit{seqCons})) \wedge$$
$$(\neg \mathit{seqCons} \leftrightarrow D_{\mathit{Threads}}(\neg \mathit{seqCons})).$$

Knowledge about Consistency

$$\models \neg \mathrm{Lin} \leftrightarrow D_{\mathrm{Threads} \uplus \{\mathrm{obs}\}}(\neg \mathrm{Lin})$$

but

$$\text{exists } E\text{: } E \models \mathrm{Lin} \wedge \neg D_{\mathrm{Threads} \uplus \{\mathrm{obs}\}}(\mathrm{Lin}))$$

Conclusion

- Ramification of consistency guarantees are notoriously oblique
- We provide declarative spec
- Uncover non-trivial relations between properties
- Previously unstudied perspective on consistency

Thank you!

Future Work

- Observational refinement
- Exploit Epistemic Logic Theory
- Other properties in Epistemic Logic
- Model-check logic to check arbitrary properties

Eventual Consistency

$$\text{alive} := \forall t \forall t' \forall id\ (\boxminus(\text{commit}(t,id) \land \Box\Diamond\ (\exists id'\ (\text{commit}(t', id'))) \rightarrow \Diamond\text{forward}(t, t', id)))$$

The observer's indistinguishability relation

$$obs(E) = \{(r,c) \in RET \times CALL \mid pos(r,E) < pos(c,E)\}$$

$$E \sim_{obs} E' \text{ :iff } obs(E) \subseteq obs(E')$$

The observer

E := (t2,call ld())(t2,ret ld(1))(t1,call st(1)) (t1,ret st(TRUE))

obs(E) = {((t2,ret ld(1)) , (t1,call st(1)))}

The observer's view is the order of non-overlapping method calls

The observer

E := (t2,call ld())(t1,call st(1)) (t2,ret ld(1)) (t1,ret st(TRUE))

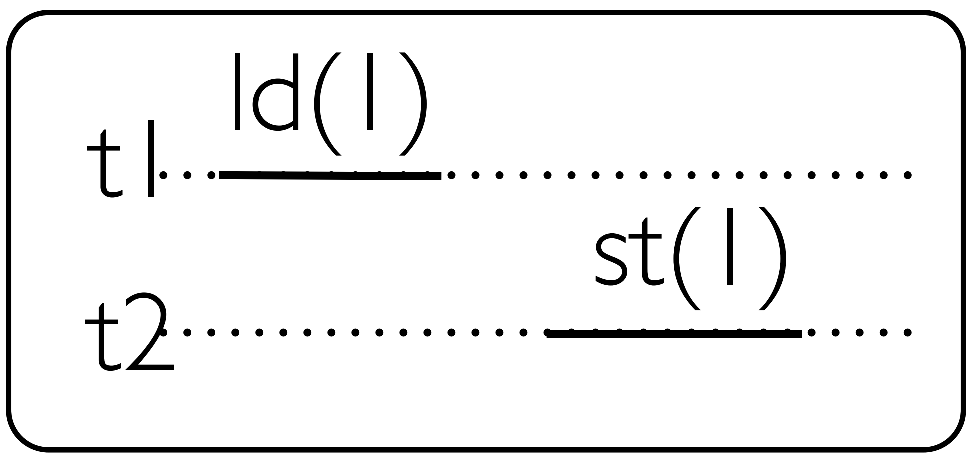
obs(E) = {}

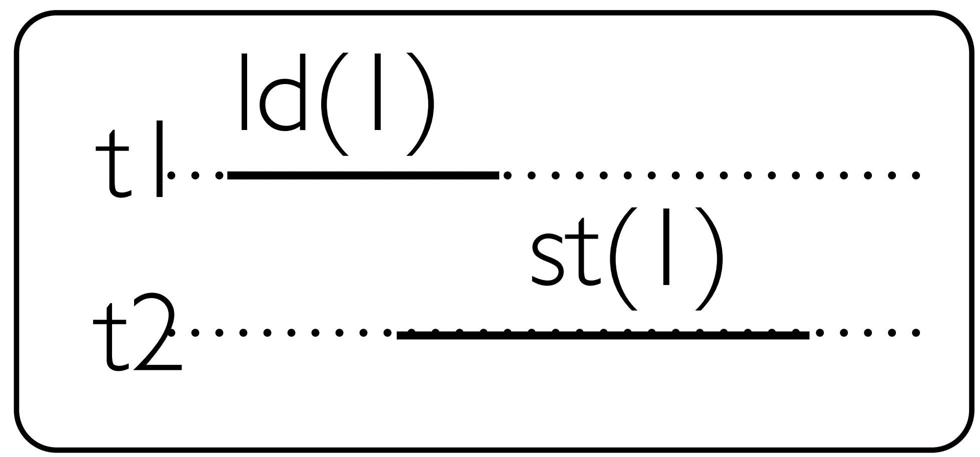
The observer's view is the order of non-overlapping method calls

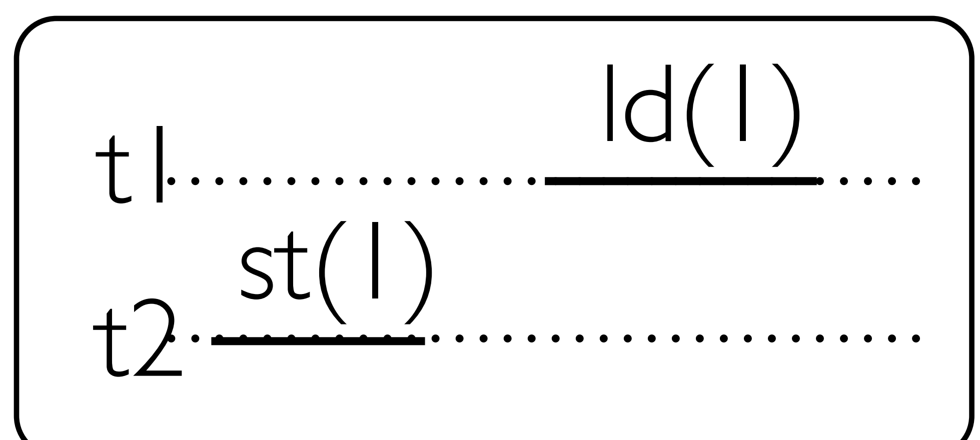
Linearizability

$$E \models \neg D_{\mathrm{THREADS} \cup \{obs\}}(\neg \mathrm{correct})$$

$$\text{:iff exists } E' \text{ s.t. } E \sim_{\mathrm{Threads} \cup \{obs\}} E' \text{ and } E' \models \mathrm{correct}$$







Syntax

In the last time-step

Until

proposition

Since

knowledge

$\Phi \ni$

$\varphi ::= p \mid \varphi \wedge \varphi \mid \neg \varphi \mid \ominus \varphi \mid \varphi S \varphi \mid \varphi U \varphi \mid D_G \varphi \mid \forall x(\varphi)$

basic

temporal

quantification

Semantics

$(E, i) \models \varphi \wedge \psi$:iff $(E, i) \models \varphi$ and $(E, i) \models \psi$

$(E, i) \models \neg\varphi$:iff not$(E, i) \models \varphi$

$(E,i) \models \ominus\varphi$:iff $i>0$ and $(E,i-1) \models \varphi$

$(E,i) \models \varphi S \psi$:iff $(E,i) \models \varphi \cup \psi$:iff

there is $j \leq i$ s.t. $(E,j) \models \psi$ and for all $j < k \leq i$: $(E,k) \models \varphi$

$(E, i) \models D_G\varphi$:iff for all (E', i'): if $(E, i) \sim_G (E', i')$ then $(E',i') \models \varphi$

$(E,i) \models \forall x(\varphi)$:iff for all $d \in D$: $(E,i) \models \varphi[d/x]$

TSO Consistency

$$E \models \neg D_{\text{THREADS}}(\neg \text{correctTSO})$$

Read from the buffer

If you read from the store buffer, you need to read the latest value stored

locallyLatest(t, a, v) := ¬(∃v′ store(t, a, v’)) S store(t, a, v))

TSO Consistency

$$
\begin{aligned}
&\mathit{correctTSO} := \\
&\quad \mathit{FlushOrder} \wedge \\
&\quad \forall t \forall a \forall v \\
&\quad (\boxminus(\mathit{load}(t,a,v) \rightarrow \\
&\quad (\mathit{ldBuff}(t,a,v) \vee \mathit{ldMem}(t,a,v))))
\end{aligned}
$$

flush in Fifo-order

"so far"

load from buffer

load from memory

Basic Predicates

E inspected at position $i \in N$

Event at position i

$(E,i) \models store(t, a,v)$:iff $E@i = (t, st(a, v))$

$(E,i) \models load(t,a,v)$:iff $E@i = (t,ld(a,v))$

$(E,i) \models flush(t,a,v)$:iff $E@i = (m_{sys},fl(t,a,v))$

We need positions for time

The memory system flushed a value

Joseph Y. Halpern and Yoram Moses. Knowledge and common knowledge in a distributed environment. Journal of the ACM, 37(3):549-587, 1990.