Trace2Consistency: A tool for automated Consistency Model Selection in Distributed Systems

Suhas Shripad Hebbar

Venkata Sai Kiran Patnaik Balivada

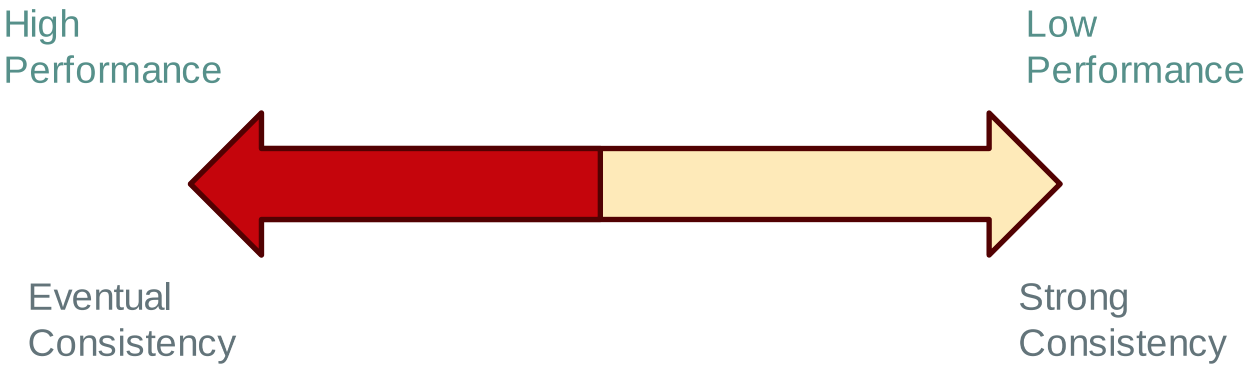
Ali Asgar Sabir

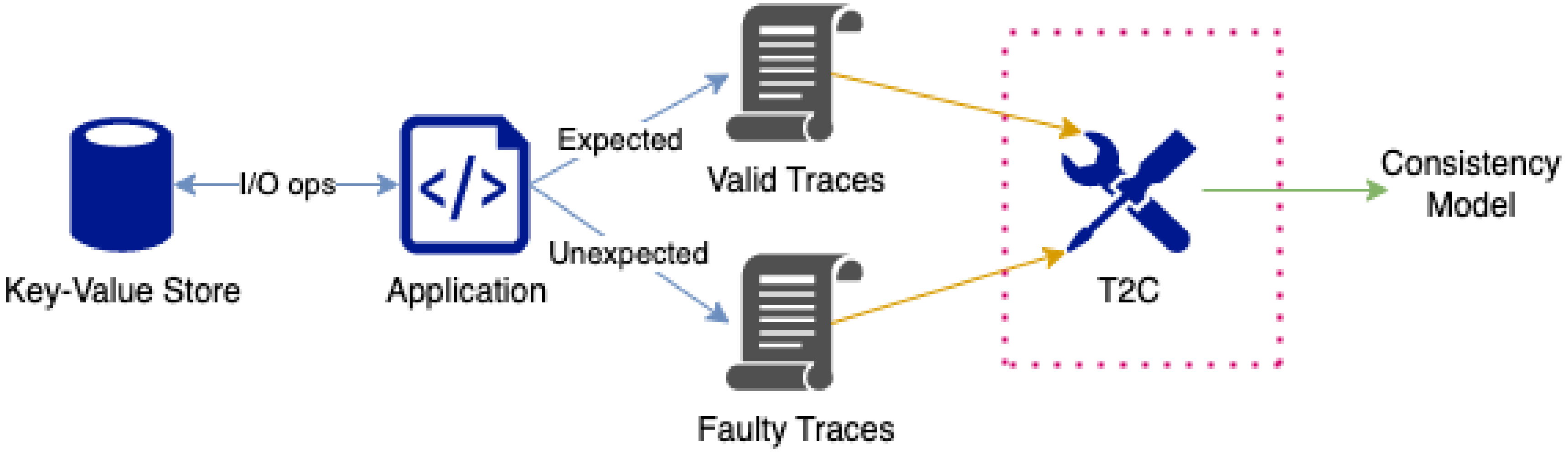
CS739 – Spring '23

Agenda

- Motivation
- Related Work
- Basic Approach
- Implementation
- Scalability

Motivation & Problem Setup





Related Work: Jepsen & Baseball

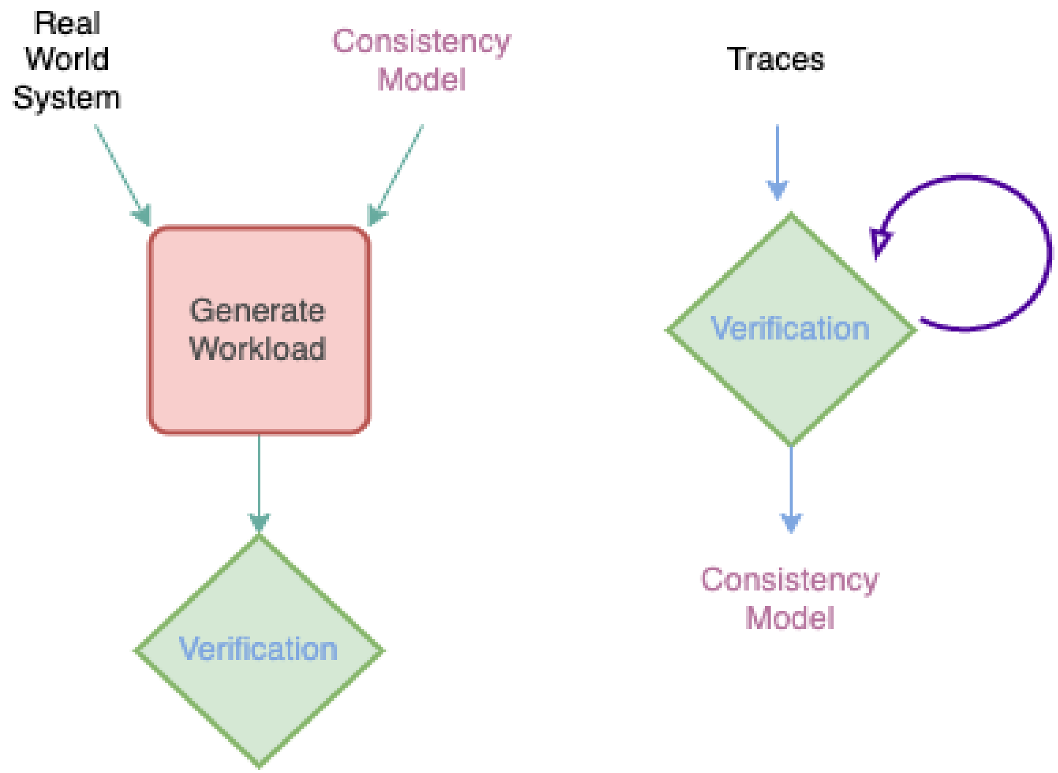


Table 1. Six consistency guarantees.

Strong Consistency	See all previous writes.
Eventual Consistency	See subset of previous writes.
Consistent Prefix	See initial sequence of writes.
Bounded Staleness	See all "old" writes.
Monotonic Reads	See increasing subset of writes.
Read My Writes	See all writes performed by reader.

Doug Terry. 2013. Replicated data consistency explained through baseball. Commun. ACM 56, 12 (December 2013), 82–89. https://doi.org/10.1145/2500500

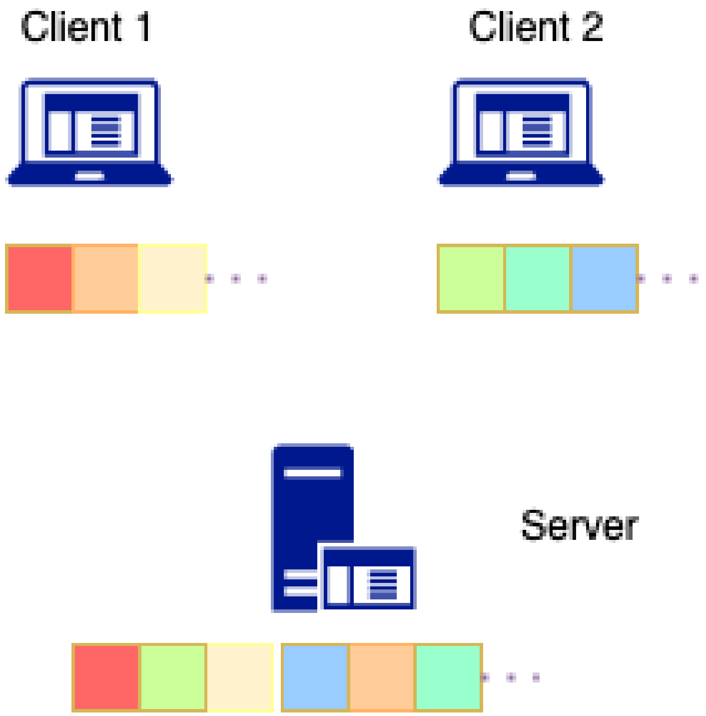
Input/Output demo

Valid	w(v1)	r(v1)	w(v2)	r(v2)
Faulty	w(v1)	w(v2)	r(v1)	r(v2)

```
Consistencies
===========
Sequential
...
```

- Demo

Basic Approach



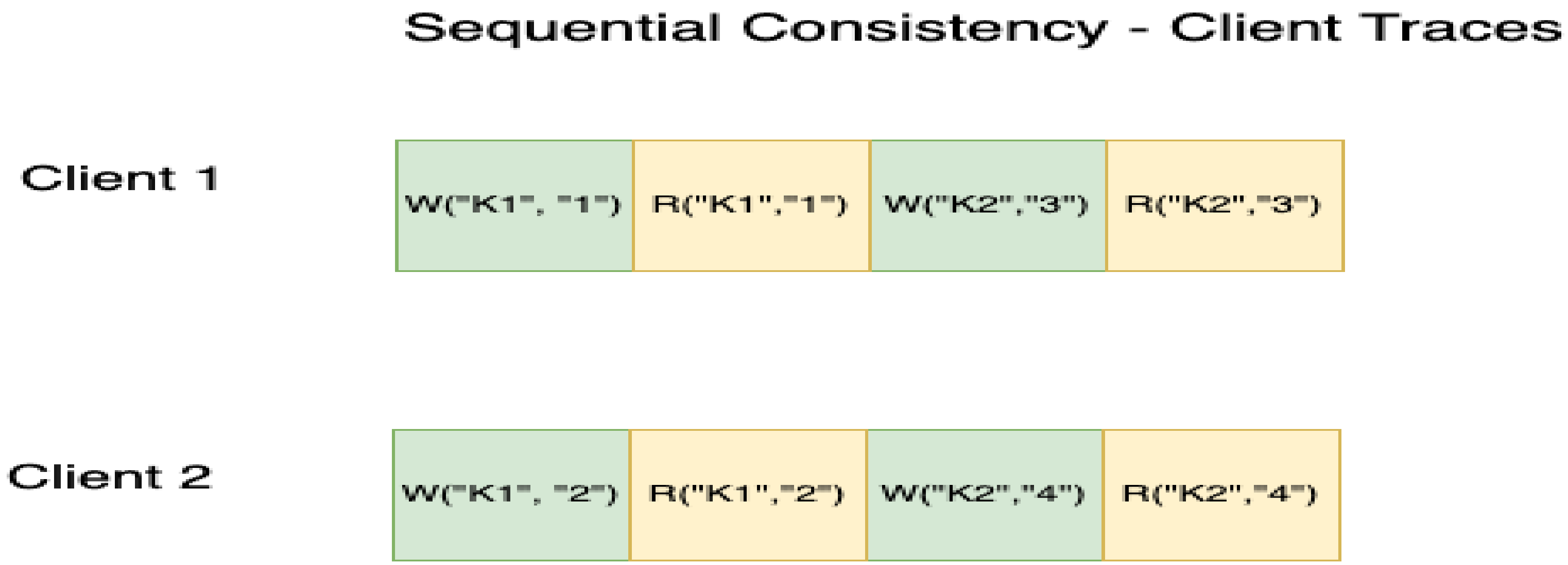
Find a schedule consistent with the client traces and allowed by the model

Verifier

- Checks what consistency models a permutation can satisfy
- The consistency models we check for
  - Sequential
  - Monotonic Reads
  - Read My Writes
  - Monotonic + Read My Writes
  - Eventual
- First checks if the permutation is valid that is Reads reflect the most recent write for an object
- After this basic check, checks the constraint for individual consistency models

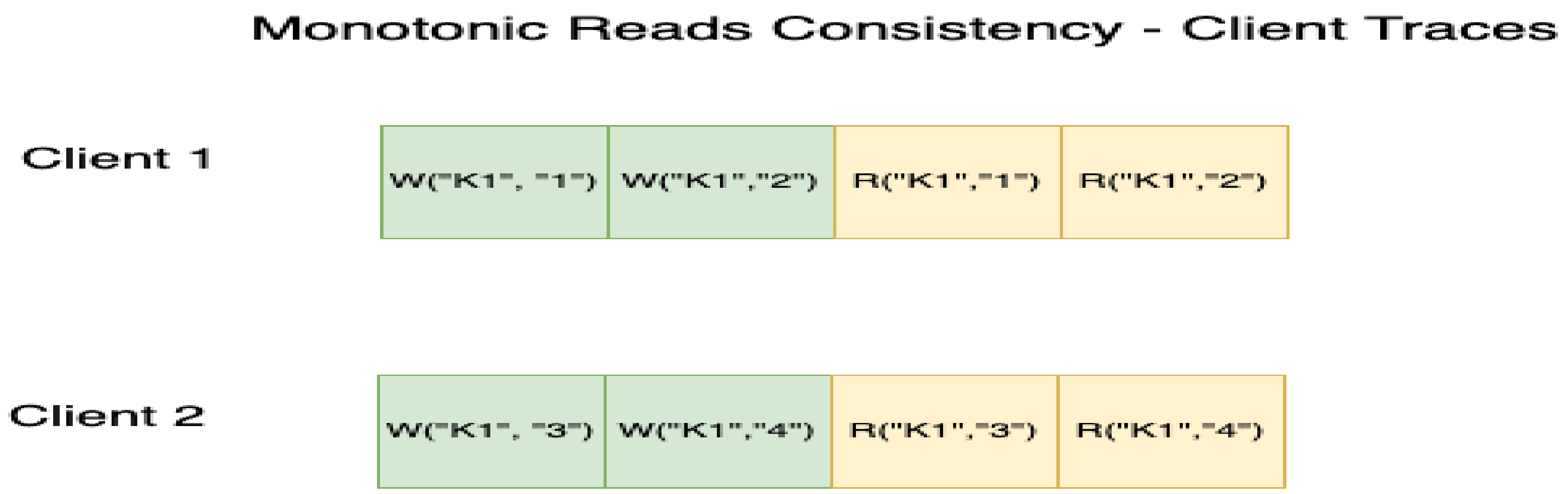
Sequential - [Demo]

- Constraints
  - Reads and writes in program order for a client
  - Across clients there can be interleaving



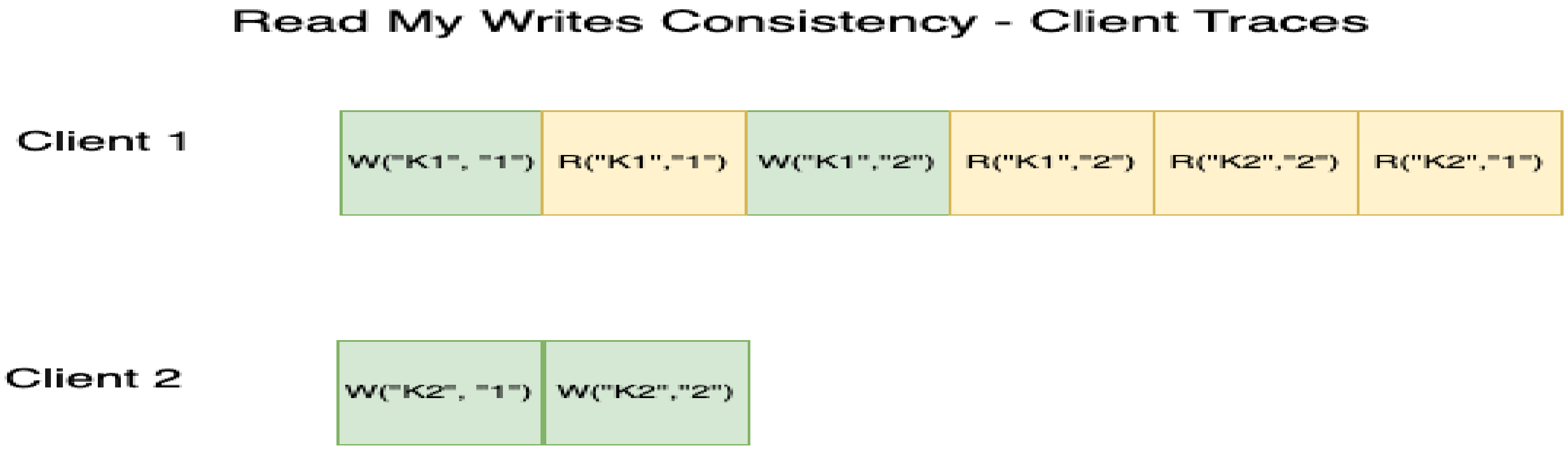
Monotonic Reads - Demo

- Constraint
  - Writes for an object cannot be reordered
  - Reads for an object cannot be reordered
  - Reads and writes can be interleaved across each other



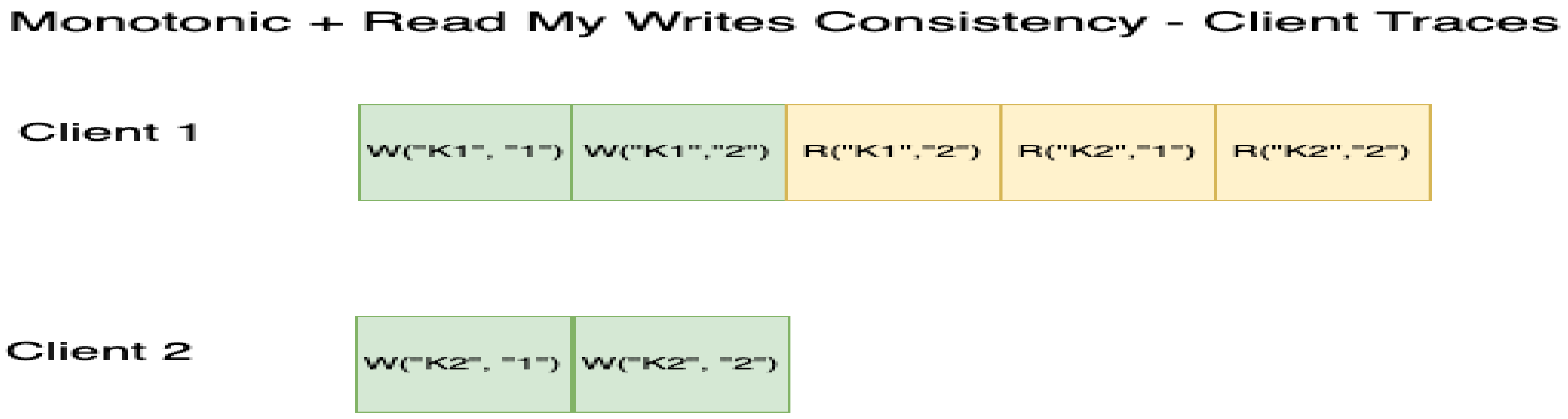
Read my writes - Demo

- Constraints
  - If a client writes a new value for a data object and then reads this object, the read will return the value that was last written by the client (or some other value that was later written by a different client)
  - For every read from a client all previous writes should occur before this read and later writes should occur after



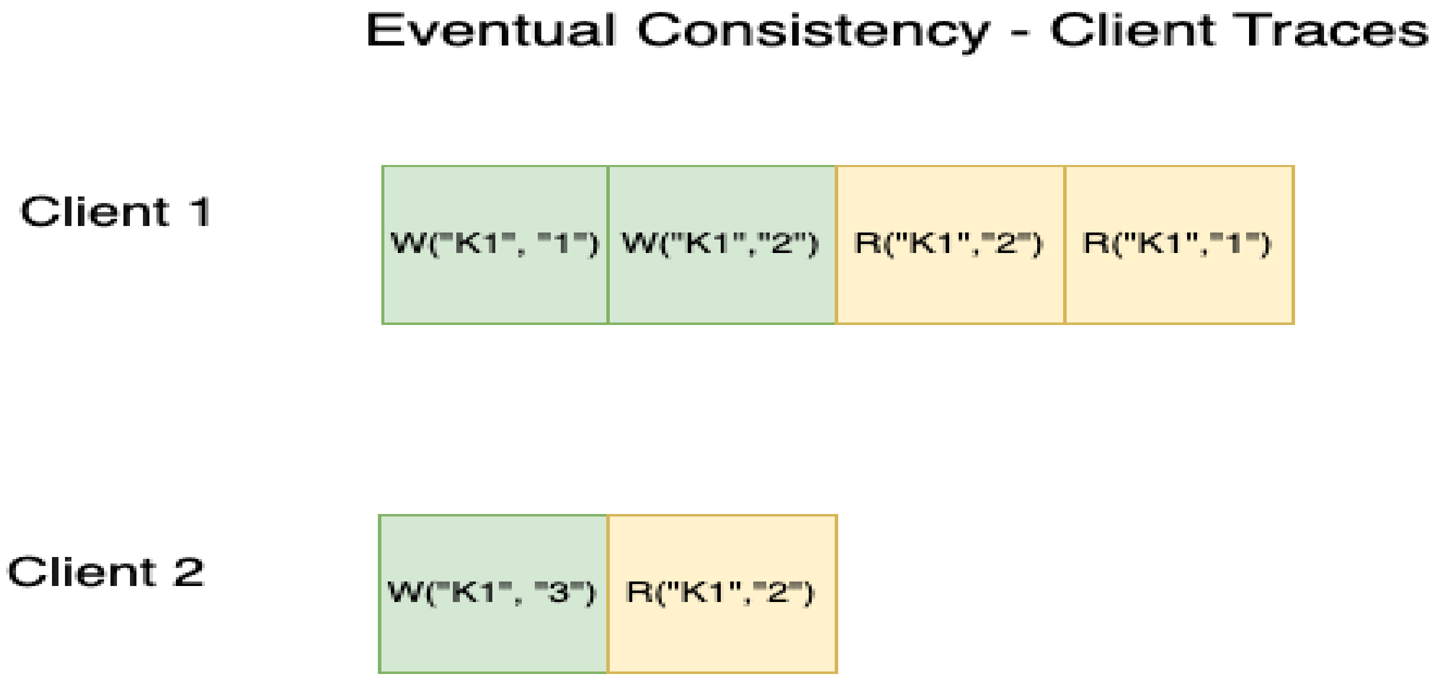
Monotonic + Read My Writes - Demo

- Constraint
  - Combined constraints of monotonic reads and read my writes
  - A client observes a data store that is consistent with its own actions



Eventual Consistency - Demo

- If the permutation is valid, it will always follow eventual consistency



SMT solver based Consistency checker

- Allow specifying constraints in an abstract manner for easier exploration.
- Variation in performance? Slower? Faster?

Implementation

- Choice of solver
  - Z3
  - Cvc5
- Interface
  - SMTLIB2
  - FFI
- Integration with existing brute force solver

Challenges

- SMT solvers can handler quantifiers but do not seem to handle them well.

```
(forall ((key String) (index Int))
…
(seq.contains keys (seq.unit key))
(>= index 1)
(< index (seq.len (select keysToOps key)))
```

- Poor visibility into internals for debugging poor performance or failure to halt by solver

Define rules iteratively

```
(let ((a!1 (seq.nth trace (seq.nth final_order 0)))
(=> (= (op a!1) Read)
      (= (select prevOp (key a!1)) (value a!1))))

(let ((a!1 (seq.nth trace (seq.nth final_order 1)))
(=> (= (op a!1) Read)
      (= (select prevOp (key a!1)) (value a!1))))

(let ((a!1 (seq.nth trace (seq.nth final_order 2)))
(=> (= (op a!1) Read)
      (= (select prevOp (key a!1)) (value a!1))))
```

https://asciinema.org/a/F3trYdH0JxBWfYpF3MftseZUx

Performance

- High overhead compared to brute force solver.
  - In the order of 1000s
  - Eg: 3.5 s vs 206 μs for trace size of 6
- Preliminary testing shows exponential growth in execution time.
- Unclear overhead for larger input sizes due to specifying rules at each index instead of leveraging quantifiers.

Conclusions

- There is a valid path to implement the constraints in an SMT solver.
- But the current approach of specifying rules iteratively may not scale for larger traces.

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON