

Sonic-Chain

*A Lock-free, Pseudo-static Approach Toward
Concurrency in Blockchains*

Kian Paimani

kpi240@student.vu.nl / kian@parity.io

@kianenigma





What is a Blockchain Anyhow?

Aside from all the 🤪

- Technical
 - ***Distributed*** network of nodes.
 - ***Distributed*** key value database.
 - ***Decentralized*** transaction processing state machine.
- Organisation
 - Trustless.
 - Not owned by a single big entity.
 - Censorship resistance.
 - Immutable.

Problem?

Blockchains are known for being **costly** *and* **slow**.

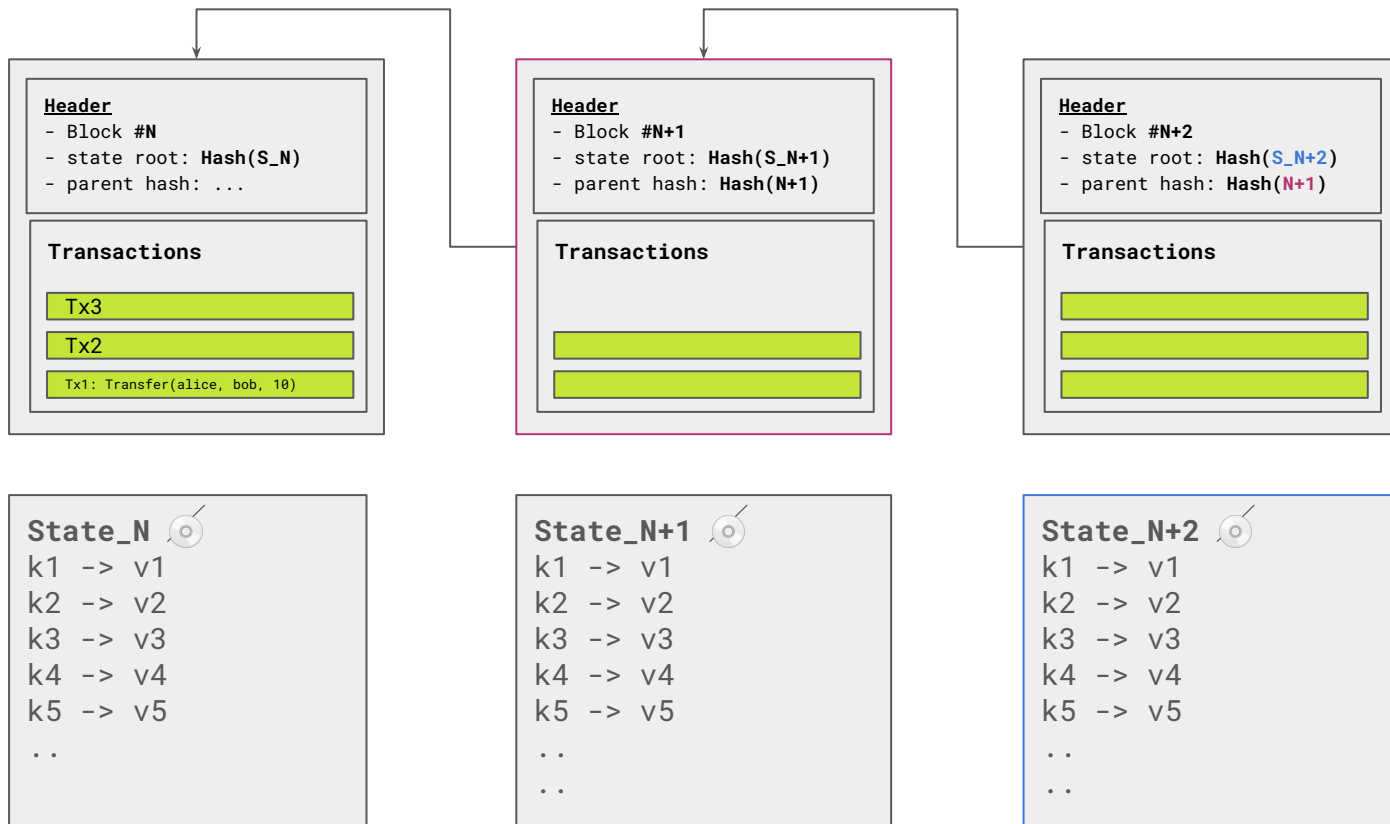
- Lack of throughput, in terms of transactions per second (tps).
- Concurrency 
 - *But how?*
 - *At what overhead cost?* 

Blockchains 101

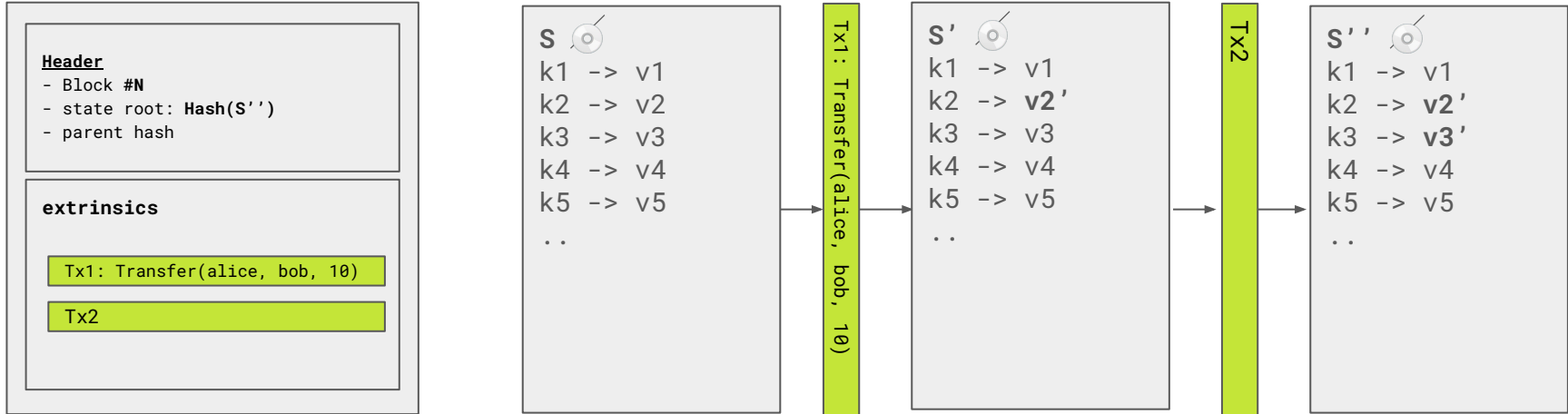
ridiculously oversimplified, but enough for us.

Blockchain

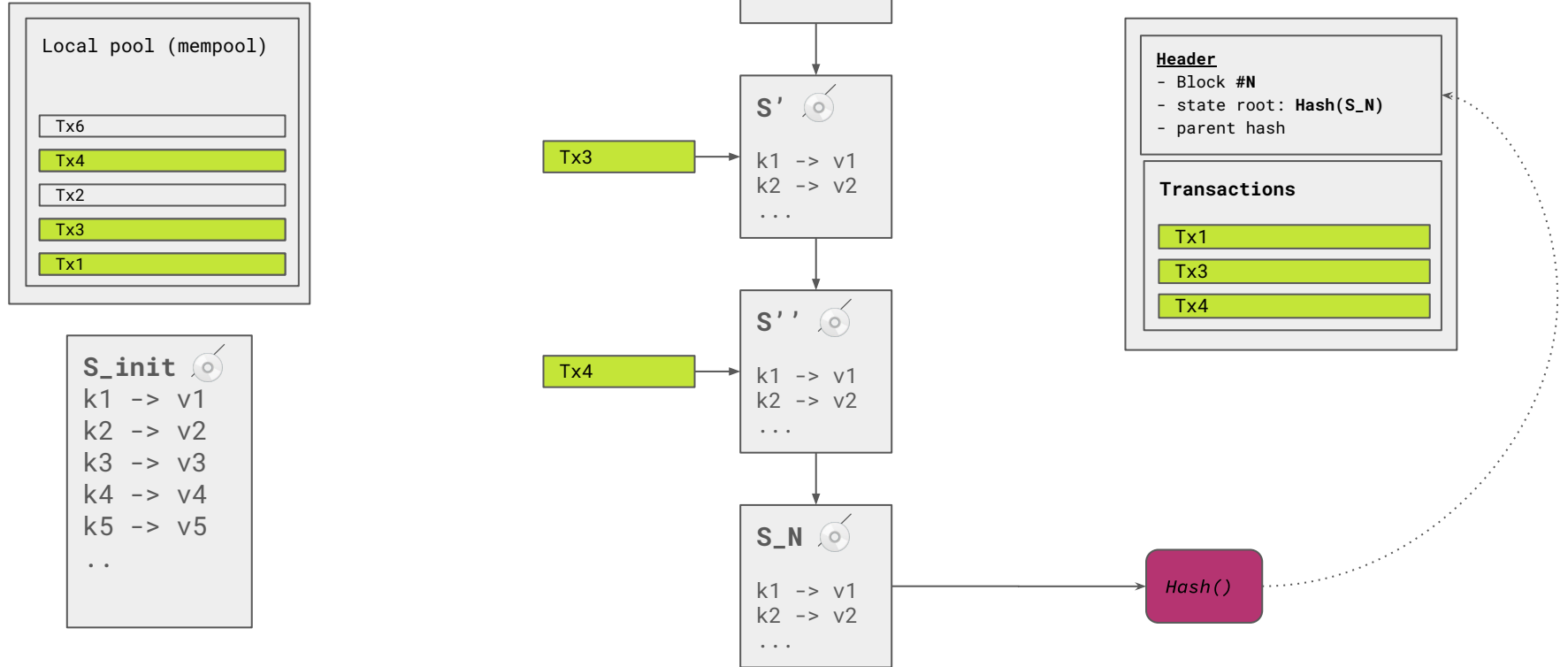
Umm.. block-chain.



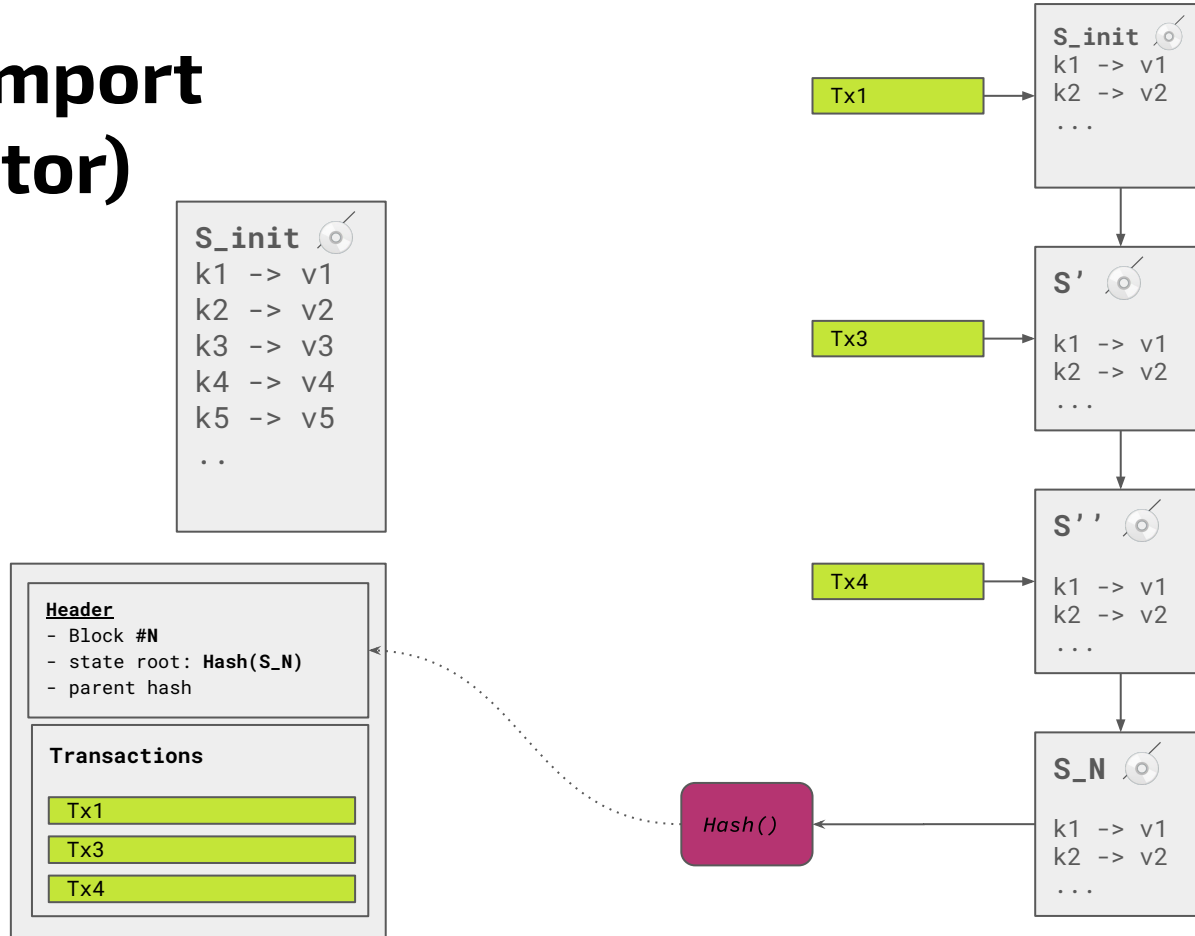
State Machine



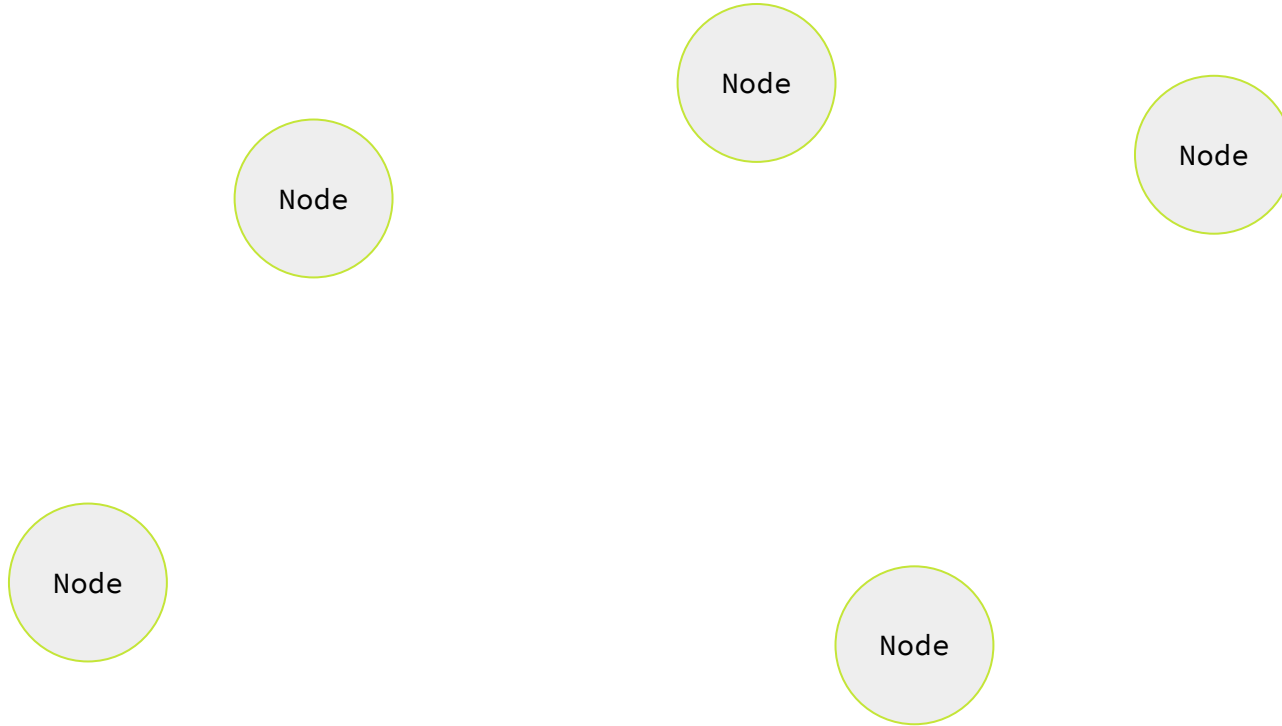
Block Authoring



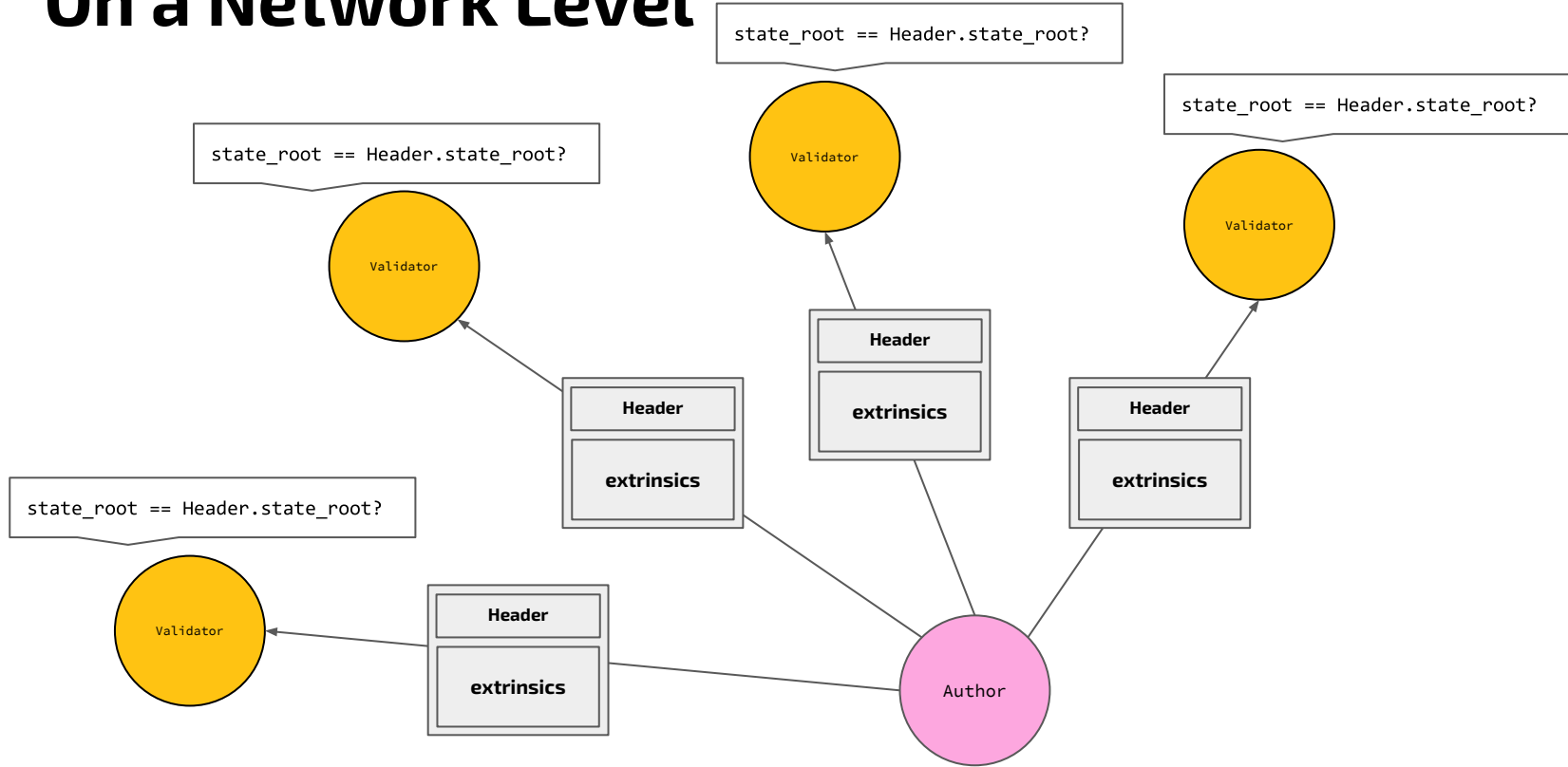
Block Import (Validator)



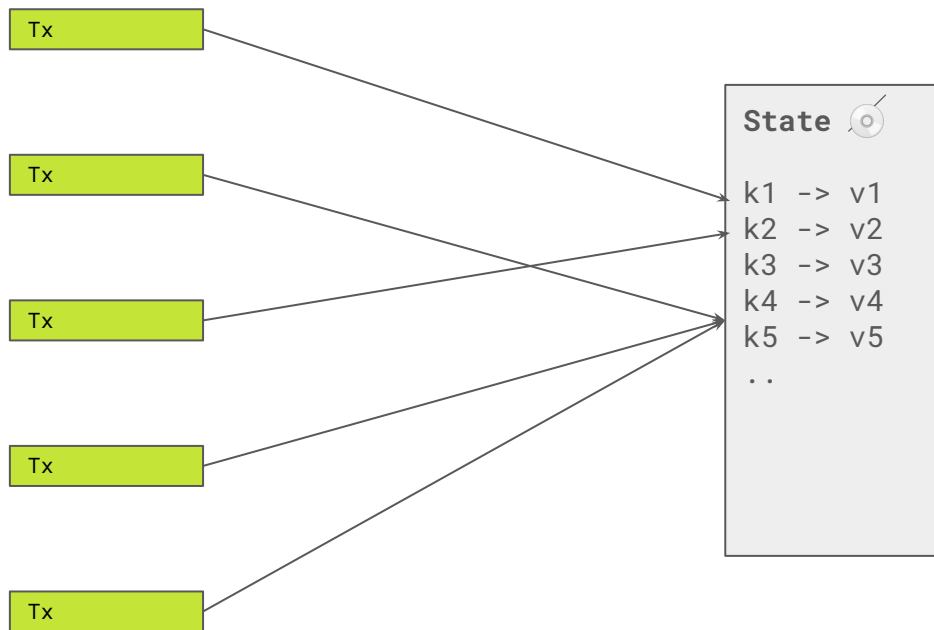
On a Network Level



On a Network Level



Concurrency? Easy.



Yes!

This is shared state concurrency.

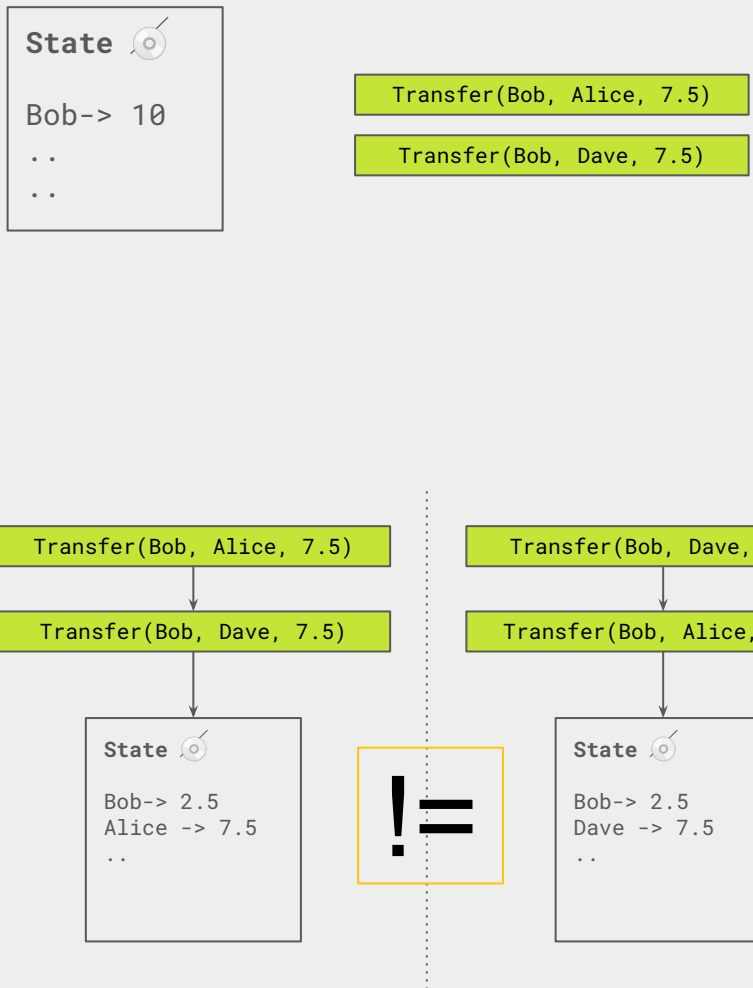
- Locks, Mutex, STM, All the good stuff (**concurrency control**).
- Database industry has already solved this.
- Problem solved.
- Thesis done.
- Have a nice day.

No

Sorry.

Let's make sure we get this right.

Concurrency Control is **NOT**
ENOUGH 💡.



Maintaining Determinism is Absolute



- Both were **valid** authoring, but not the **same**.
- Locking, STM, and generally **concurrency control** mechanism are non deterministic.

Well, Someone Must Have Solved This.



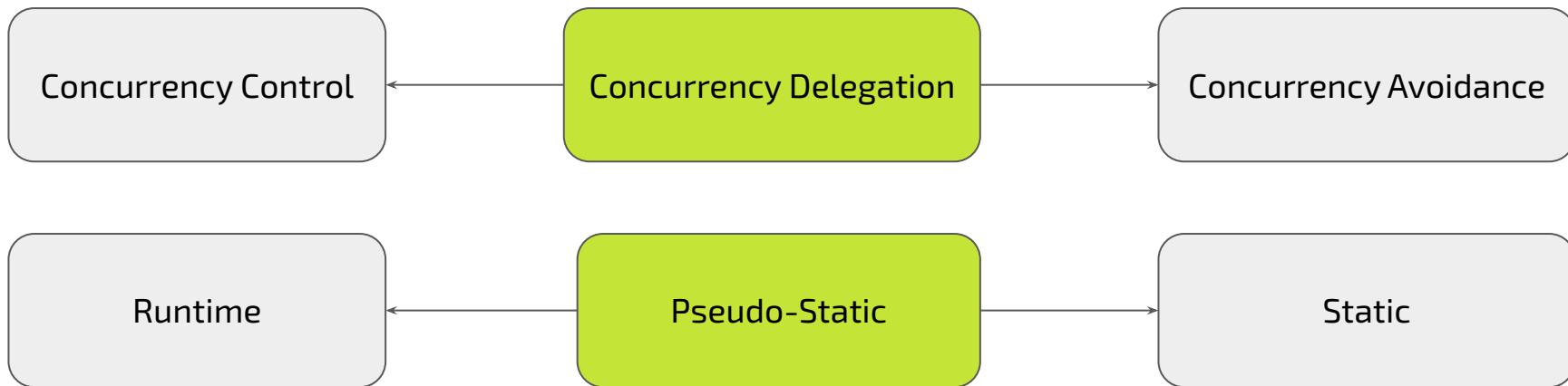
- **Concurrency control:** Track a *runtime* dependency graph.
- **Concurrency avoidance:**
Executes as much as you can concurrently, anything that conflicts -> Second sequential phase.
- **Static Analysis:** new programming language, used used on combination.

Downsides

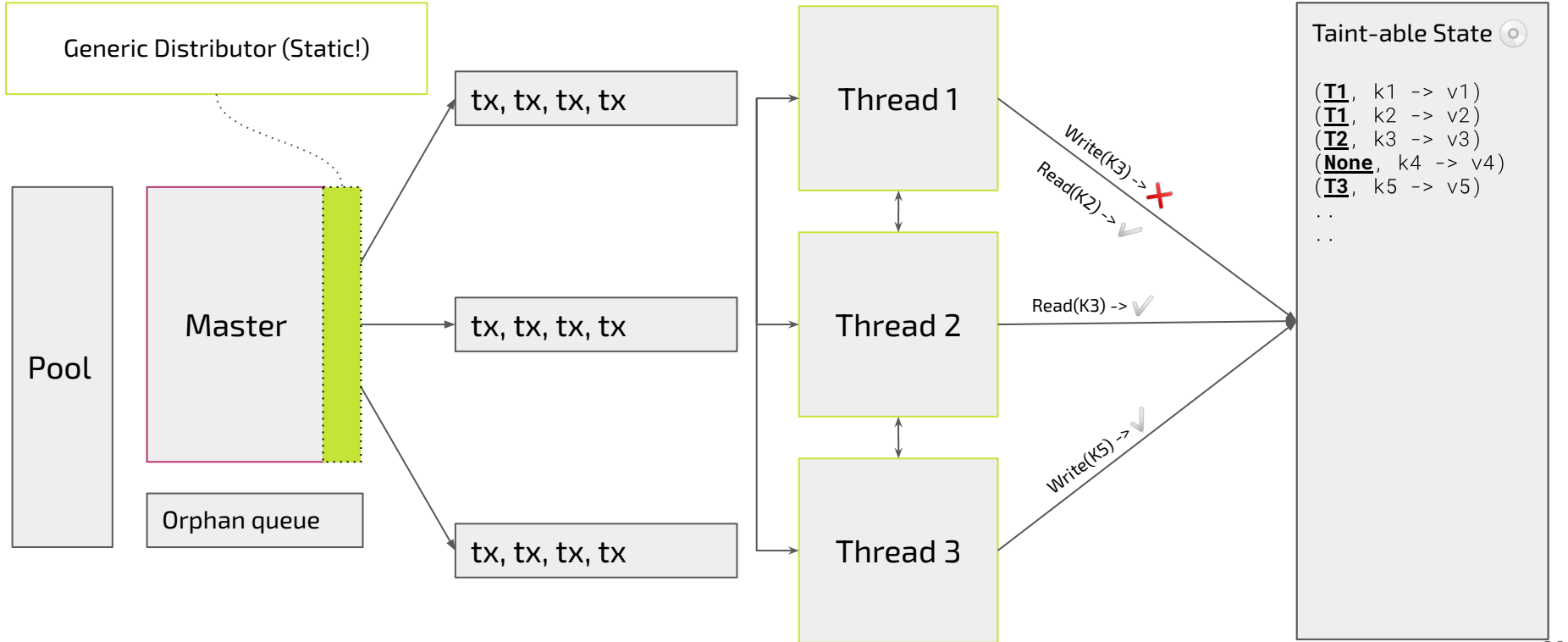


- Runtime Overhead
- Validation Overhead
 - That dependency graph thingy must be propagated - $O(N)$
 - Validators need to parse and use it.
- (Static) Radical change to a programming model.

A New Balance in the Spectrum.



Concurrency Delegation



Static Hints

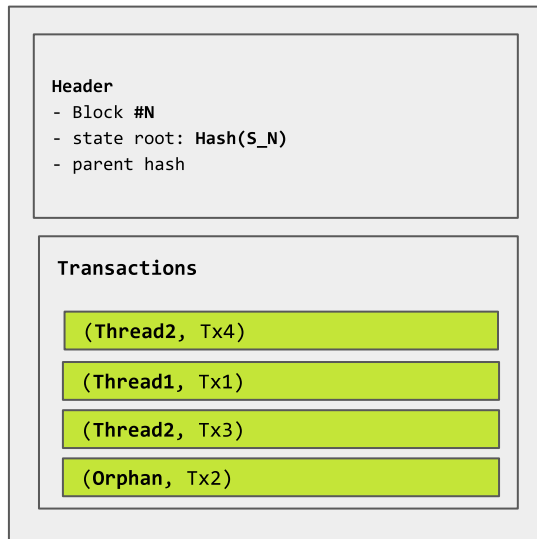
- End of the day: ***We want to know the storage access of each transaction.***
- A transaction's code can already hint a lot at its storage access.
- Static w.r.t. to the ***lifetime of the transaction*** itself!
 - Of course you **cannot** execute a transaction to make a guess here.



```
#[access = (transactor, arg1)]  
fn transaction(transactor, arg1, arg2, ...) {  
    /* ... */  
}
```

The Outcome

- We can't rely on the static hints, we need some *minimal* runtime backup.
- Wait free.
- Minimal overhead on block size and validation process.



Lots of other details

That we skip.

Let us instead talk numbers .

Transfer

```
#[access = (|origin|
    vec![<BalanceOf<R>>::key_for(origin) <BalanceOf<R>>::key_for(dest.clone())]
)]
fn transfer(runtime, origin, dest: AccountId, value: Balance) {
    // If we fail at this step, it is fine. We have not written anything yet.
    let mut old_balance =
        BalanceOf::read(runtime, origin).or_forward()?;

    if let Some(remaining) = old_balance.free.checked_sub(value) {
        // update origin. Failure is okay.
        old_balance.free = remaining;

        BalanceOf::write(runtime, origin, old_balance)
            .expect("Origin's balance key must be owned by the current thread.");

        // update dest.
        BalanceOf::mutate(runtime, dest, |old| old.free += value).or_orphan()?;

        Ok(())
    } else {
        Err(DispatchError::LogicError("Does not have enough funds."))
    }
}
```


Millionaires Playground

- A number of accounts
- A number of random transfers between them.
- All accounts have a gazillion balance. All transfers are Ok.
- We only measure the execution of the **authoring** and **validation** process. Pool processing, distribution, connected components etc. are all ***pre-processing***.
- Experiments:
 - Sequential.
 - Concurrency Delegation + Round Robin distribution (***blid***).
 - Concurrency Delegation + Connected Components distribution, (***using static hints***).
- Target machine: My humble MBP
 - 2,3 GHz 8-Core Intel Core i9
 - 32 GB 2400 MHz DDR4

Results



millionaires_playground	-	-	-	-
type	members	transactions	authoring time(ms)	validation time(ms)
Sequential	1000	250	1902	1809
Sequential	1000	500	3714	3814
Sequential	1000	1000	7642	7641
Sequential	1000	2000	15267	15209
Concurrent(RoundRobin-4)	1000	250	884	682
Concurrent(RoundRobin-4)	1000	500	2249	1872
Concurrent(RoundRobin-4)	1000	1000	5513	4623
Concurrent(RoundRobin-4)	1000	2000	12705	10719
Concurrent(ConnectedComponents-4)	1000	250	633	469
Concurrent(ConnectedComponents-4)	1000	500	1239	923
Concurrent(ConnectedComponents-4)	1000	1000	9368	7081
Concurrent(ConnectedComponents-4)	1000	2000	19148	14827

Logs tell us a lot

Like, a *lot*.

Why RR Fails.



```
2020-10-05T11:23:17 Info benchmarks [main (ThreadId(1))] - 🖨 Generating millionaires_playground(1000, 2000)
2020-10-05T11:23:22 Info concurrent-exec [main (ThreadId(1))] - 📖 Authoring block with 2000 transactions.
```

```
2020-10-05T11:23:25 Info master [main (ThreadId(1))] - Finishing Collection phase with [505 executed][280 forwarded][1215 orphaned]
```

```
2020-10-05T11:23:34 Warn exec [main (ThreadId(1))] - ⌚ authoring took 12.494746267s
```

Why CC Succeeds (sometimes).



```
2020-10-05T11:23:52 Info concurrent-exec [main (ThreadId(1))] - 📖 Authoring block with 250 transactions.
```

```
2020-10-05T11:23:52 Info tx-dist [main (ThreadId(1))] - Assigning outcome = [(18, 63), (19, 63), (20, 62), (21, 62)]
```

```
2020-10-05T11:23:53 Info master [main (ThreadId(1))] - Finishing Collection phase with [250 executed][0 forwarded][0 orphaned]
```

```
2020-10-05T11:23:53 Warn exec [main (ThreadId(1))] - ⌚ authoring took 594.29126ms
```

Why CC Also Fails (sometimes).



```
2020-10-05T11:24:10 Info concurrent-exec [main (ThreadId(1))] - 📖 Authoring block with 1000 transactions.
```

```
2020-10-05T11:24:10 Info tx-dist [main (ThreadId(1))] - Assigning outcome = [(26, 951), (27, 17), (28, 16), (29, 16)]
```

```
2020-10-05T11:24:19 Info master [main (ThreadId(1))] - Finishing Collection phase with [1000 executed][0 forwarded][0 orphaned]
```

```
2020-10-05T11:24:19 Warn exec [main (ThreadId(1))] - ⌚ authoring took 9.267242013s
```

Final Words 🧐

- Validation overhead is minimal to zero, it is always faster.
- Number of forwarded transactions is key.
- Good initial hint is key.
- Punch line:

*With **reasonably accurate** static hints, we can achieve a **near-ideal gain in throughput** of transactions, whilst not needing a complicated concurrency control mechanism. Most importantly, because of the trivial validation task, our approach is **deterministic** out of the box.*



growing_economy	-	-	-	-
type	members	transactions	authoring time(ms)	validation time(ms)
Sequential	1000	2000	15077	14701
Sequential	2000	2000	14768	14835
Sequential	3000	2000	14542	14669
Sequential	4000	2000	14965	14905
Concurrent(RoundRobin-4)	1000	2000	12913	10885
Concurrent(RoundRobin-4)	2000	2000	11029	9166
Concurrent(RoundRobin-4)	3000	2000	9690	7928
Concurrent(RoundRobin-4)	4000	2000	8754	7243
Concurrent(ConnectedComponents-4)	1000	2000	19529	14791
Concurrent(ConnectedComponents-4)	2000	2000	18738	14209
Concurrent(ConnectedComponents-4)	3000	2000	11885	8882
Concurrent(ConnectedComponents-4)	4000	2000	4961	3707



millionaires_playground_2key	-	-	-	-
type	members	transactions	authoring time(ms)	validation time(ms)
Concurrent(ConnectedComponents-4)	1000	250	594	463
Concurrent(ConnectedComponents-4)	1000	500	1229	916
Concurrent(ConnectedComponents-4)	1000	1000	9267	6879
Concurrent(ConnectedComponents-4)	1000	2000	19374	14464
millionaires_playground_1key	-	-	-	-
type	members	transactions	authoring time(ms)	validation time(ms)
Concurrent(ConnectedComponents-4)	1000	250	963	793
Concurrent(ConnectedComponents-4)	1000	500	2100	1711
Concurrent(ConnectedComponents-4)	1000	1000	5377	4549
Concurrent(ConnectedComponents-4)	1000	2000	12063	10375



```
2020-10-05T14:08:01 Info concurrent-exec [main (ThreadId(1))] - Authoring block with 250 transactions.
2020-10-05T14:08:01 Info master [main (ThreadId(1))] - Broadcasting Message { payload: Task(Authoring), from: 1 }
2020-10-05T14:08:01 Info worker [Worker#0 (ThreadId(18))] - Received task Task(Authoring).
2020-10-05T14:08:01 Info worker [Worker#1 (ThreadId(19))] - Received task Task(Authoring).
2020-10-05T14:08:01 Info worker [Worker#2 (ThreadId(20))] - Received task Task(Authoring).
2020-10-05T14:08:01 Info worker [Worker#3 (ThreadId(21))] - Received task Task(Authoring).
2020-10-05T14:08:01 Info tx-dist [main (ThreadId(1))] - Assigning outcome = [(18, 63), (19, 63), (20, 62), (21, 62)]
2020-10-05T14:08:01 Info master [main (ThreadId(1))] - Broadcasting Message { payload: TransactionDistributionDone, from: 1 }
2020-10-05T14:08:02 Info worker [Worker#1 (ThreadId(19))] - Sending report Message { payload: AuthoringReport(43, 20), from: 19
}. From 43 executed, 43 were ok and 0 were logic error.
2020-10-05T14:08:02 Info worker [Worker#3 (ThreadId(21))] - Sending report Message { payload: AuthoringReport(45, 17), from: 21
}. From 45 executed, 45 were ok and 0 were logic error.
2020-10-05T14:08:02 Info worker [Worker#2 (ThreadId(20))] - Sending report Message { payload: AuthoringReport(47, 15), from: 20
}. From 47 executed, 47 were ok and 0 were logic error.
2020-10-05T14:08:02 Info worker [Worker#0 (ThreadId(18))] - Sending report Message { payload: AuthoringReport(56, 7), from: 18 }.
From 56 executed, 56 were ok and 0 were logic error.
2020-10-05T14:08:02 Info master [main (ThreadId(1))] - Finishing Collection phase with [191 executed][14 forwarded][45 orphaned]
2020-10-05T14:08:02 Info master [main (ThreadId(1))] - Broadcasting Message { payload: TaskDone, from: 1 }
2020-10-05T14:08:02 Info master [main (ThreadId(1))] - Starting orphan phase with 45 transactions.
2020-10-05T14:08:02 Debug worker [Worker#0 (ThreadId(18))] - Worker thread task loop started. Going to park until task.
2020-10-05T14:08:02 Debug worker [Worker#2 (ThreadId(20))] - Worker thread task loop started. Going to park until task.
2020-10-05T14:08:02 Debug worker [Worker#3 (ThreadId(21))] - Worker thread task loop started. Going to park until task.
2020-10-05T14:08:02 Debug worker [Worker#1 (ThreadId(19))] - Worker thread task loop started. Going to park until task.
2020-10-05T14:08:02 Info master [main (ThreadId(1))] - Orphan pool execution outcome: 45 ok, 0 logical error.
2020-10-05T14:08:02 Warn exec [main (ThreadId(1))] - ⌚ authoring took 963.528756ms
```



```
2020-10-05T14:08:39 Info concurrent-exec [main (ThreadId(1))] - Authoring block with 2000 transactions.
2020-10-05T14:08:40 Info master [main (ThreadId(1))] - Broadcasting Message { payload: Task(Authoring), from: 1 }
2020-10-05T14:08:40 Info worker [Worker#0 (ThreadId(30))] - Received task Task(Authoring).
2020-10-05T14:08:40 Info worker [Worker#1 (ThreadId(31))] - Received task Task(Authoring).
2020-10-05T14:08:40 Info worker [Worker#3 (ThreadId(33))] - Received task Task(Authoring).
2020-10-05T14:08:40 Info worker [Worker#2 (ThreadId(32))] - Received task Task(Authoring).
2020-10-05T14:08:40 Info tx-dist [main (ThreadId(1))] - Assigning outcome = [(30, 500), (31, 500), (32, 500), (33, 500)]
2020-10-05T14:08:40 Info master [main (ThreadId(1))] - Broadcasting Message { payload: TransactionDistributionDone, from: 1 }
2020-10-05T14:08:42 Info worker [Worker#2 (ThreadId(32))] - Sending report Message { payload: AuthoringReport(181, 319), from: 32
}. From 181 executed, 181 were ok and 0 were logic error.
2020-10-05T14:08:42 Info worker [Worker#1 (ThreadId(31))] - Sending report Message { payload: AuthoringReport(185, 315), from: 31
}. From 185 executed, 185 were ok and 0 were logic error.
2020-10-05T14:08:42 Info worker [Worker#3 (ThreadId(33))] - Sending report Message { payload: AuthoringReport(185, 315), from: 33
}. From 185 executed, 185 were ok and 0 were logic error.
2020-10-05T14:08:42 Info worker [Worker#0 (ThreadId(30))] - Sending report Message { payload: AuthoringReport(186, 314), from: 30
}. From 186 executed, 186 were ok and 0 were logic error.
2020-10-05T14:08:43 Info master [main (ThreadId(1))] - Finishing Collection phase with [737 executed][119 forwarded][1144
orphaned]
2020-10-05T14:08:43 Info master [main (ThreadId(1))] - Broadcasting Message { payload: TaskDone, from: 1 }
2020-10-05T14:08:43 Info master [main (ThreadId(1))] - Starting orphan phase with 1144 transactions.
2020-10-05T14:08:43 Debug worker [Worker#3 (ThreadId(33))] - Worker thread task loop started. Going to park until task.
2020-10-05T14:08:43 Debug worker [Worker#2 (ThreadId(32))] - Worker thread task loop started. Going to park until task.
2020-10-05T14:08:43 Debug worker [Worker#0 (ThreadId(30))] - Worker thread task loop started. Going to park until task.
2020-10-05T14:08:43 Debug worker [Worker#1 (ThreadId(31))] - Worker thread task loop started. Going to park until task.
2020-10-05T14:08:52 Info master [main (ThreadId(1))] - Orphan pool execution outcome: 1144 ok, 0 logical error.
2020-10-05T14:08:52 Warn exec [main (ThreadId(1))] - ⌚ authoring took 12.063216963s
```

Complicated

Punch line is that the business of static annotation can be complicated and could be its own mini-thesis/project.