XTREEMFS

FaTLease: Scalable Fault-Tolerant Lease Negotiation with Paxos

Björn Kolbeck
Jan Stender

Zuse Institute Berlin

Motivation

- Fault-tolerance in distributed systems is difficult to ensure because...
  - we cannot distinguish between network splits, message loss and host failures
  - we cannot decide if a host is simply busy or has crashed
  - messages can be delayed arbitrarily
- Exclusive access to a shared resource must be coordinated in distributed systems
  - exclusive write lock on a file
  - cache consistency
  - election of master and slaves for database replication

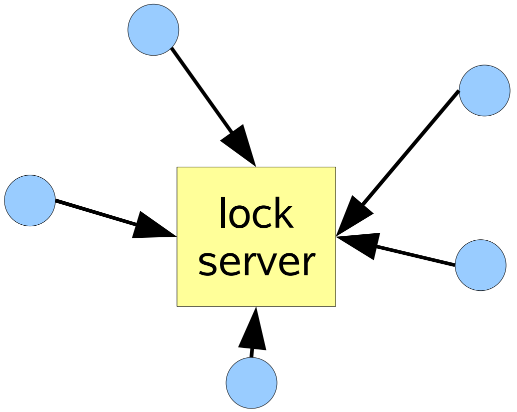
Motivation

- Leases solve the problem by granting exclusive access to the owner of a lease for a *limited period of time*

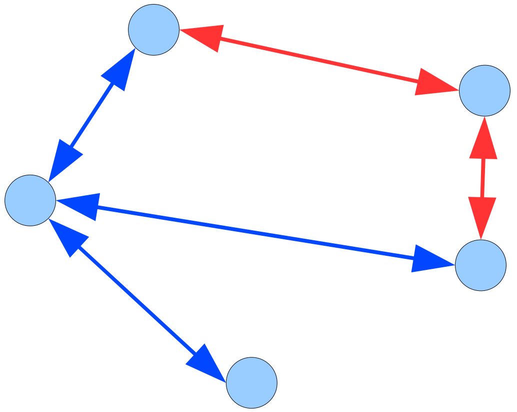
- since leases simply time out, they are particularly useful for distributed systems
  - no need for revocation
  - no need for failure detectors (busy or crashed?)
  - fail-over can be implemented easily

Motivation

- How to 'issue' leases in distributed systems =
  How to guarantee there is at most one valid lease
  - a centralized lock server
    - easy to implement
      (simple key-value database)
    - bottleneck and potential SPOF



  - or in a purely distributed fashion among hosts
    - good scalability and failure-tolerance
    - requires a mechanism to guarantee exclusivness

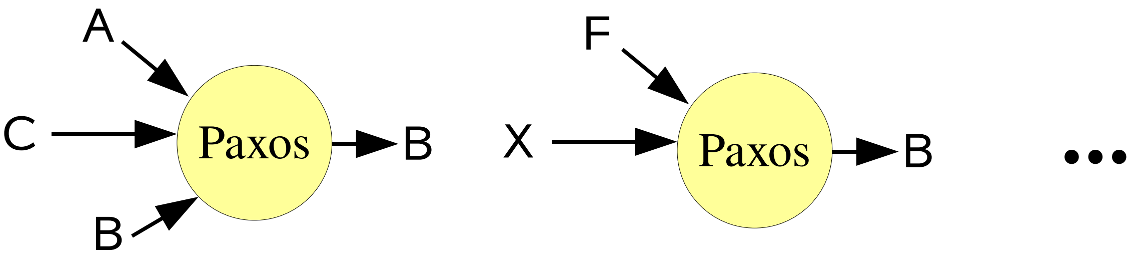


Outline

- Leases and Paxos
- FaTLease
- Evaluation of Scalability
- Conclusion

Leases and Paxos

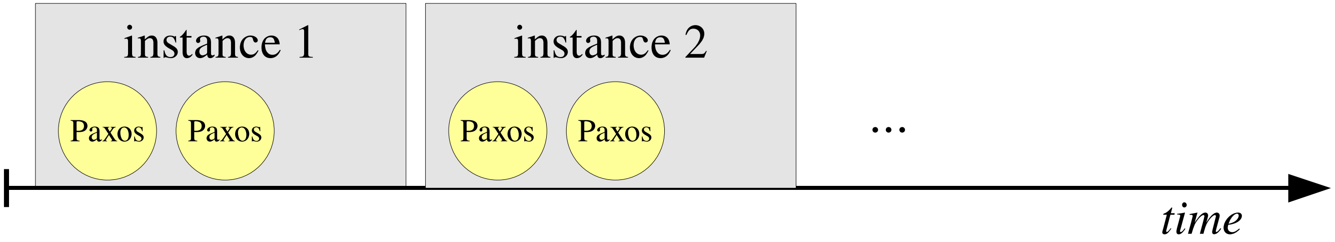
- Distributed consensus (very informal): from a set of proposed input values, a single output value is returned by all processes
- The Paxos algorithm is a fault-tolerant implementation of distributed consensus.
- Paxos can be used to agree on the owner and timeout of a lease.



- Additional mechanisms are required to issue more than one lease.

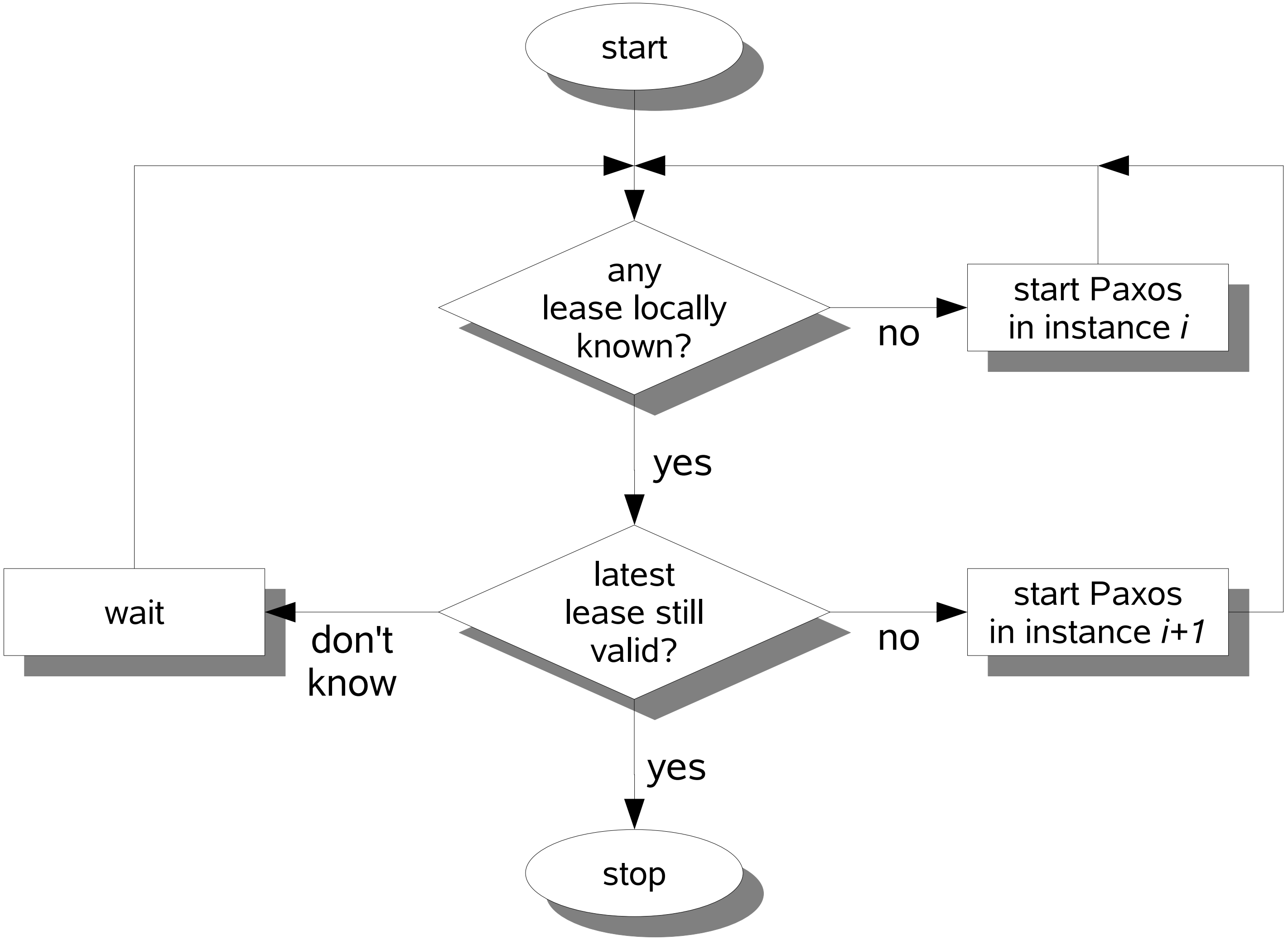
FaTLease

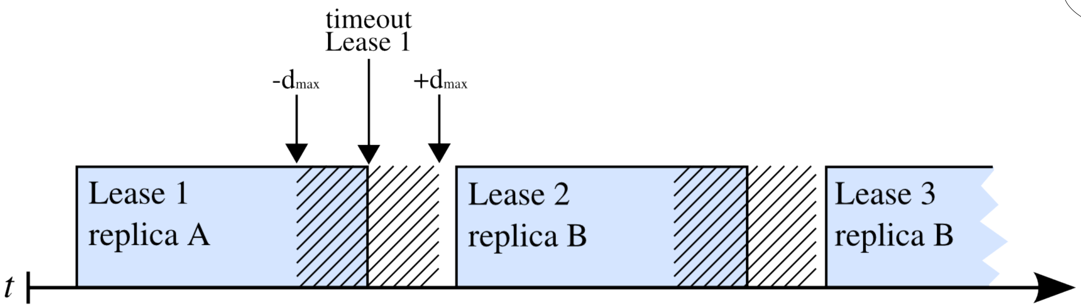
- Multipaxos: Use one instance of Paxos for each value to agree upon.



- For each lease we use Paxos to agree on the lease owner and the timeout
- Each instance is valid as until the lease timeout
- exclusiveness of the lease =
  only one valid instance at any point in time

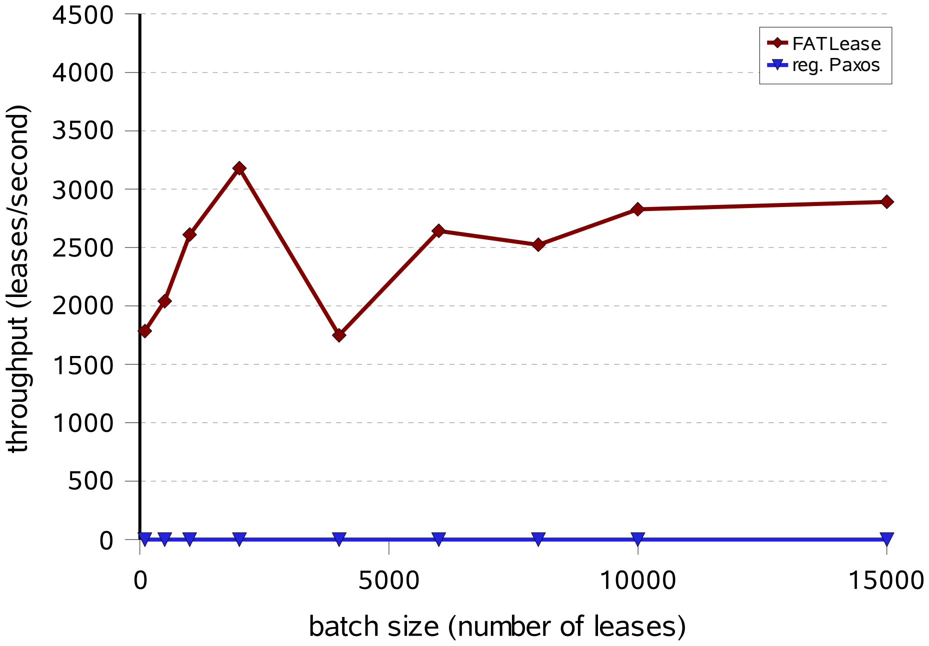
FaTLease: How it works


start
any lease locally known?
no
start Paxos in instance *i*
yes
latest lease still valid?
don't know
wait
no
start Paxos in instance *i*+*1*
yes
stop



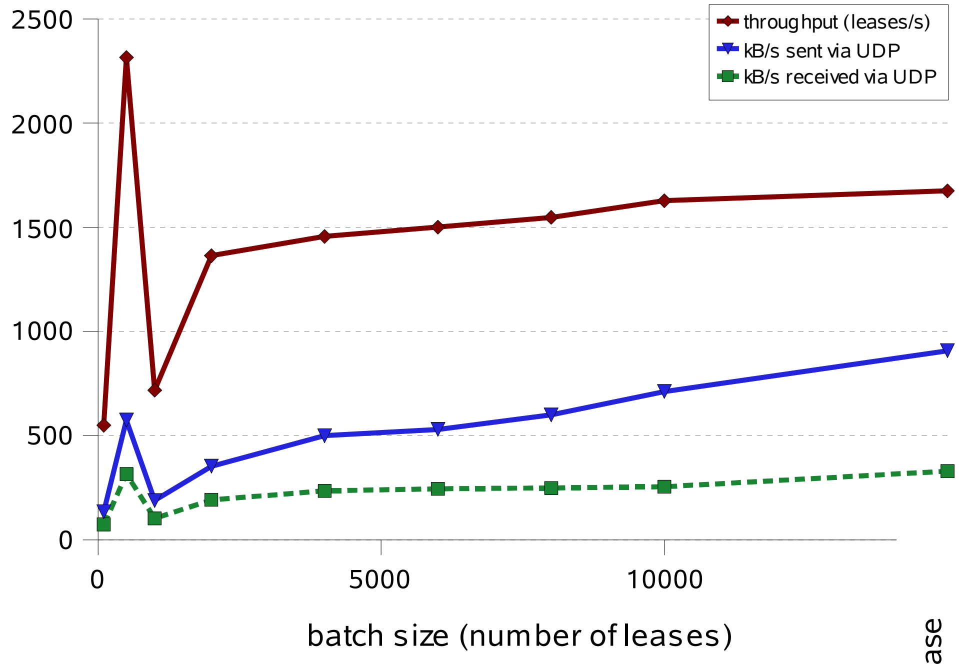
FaTLease: No need for persistent state

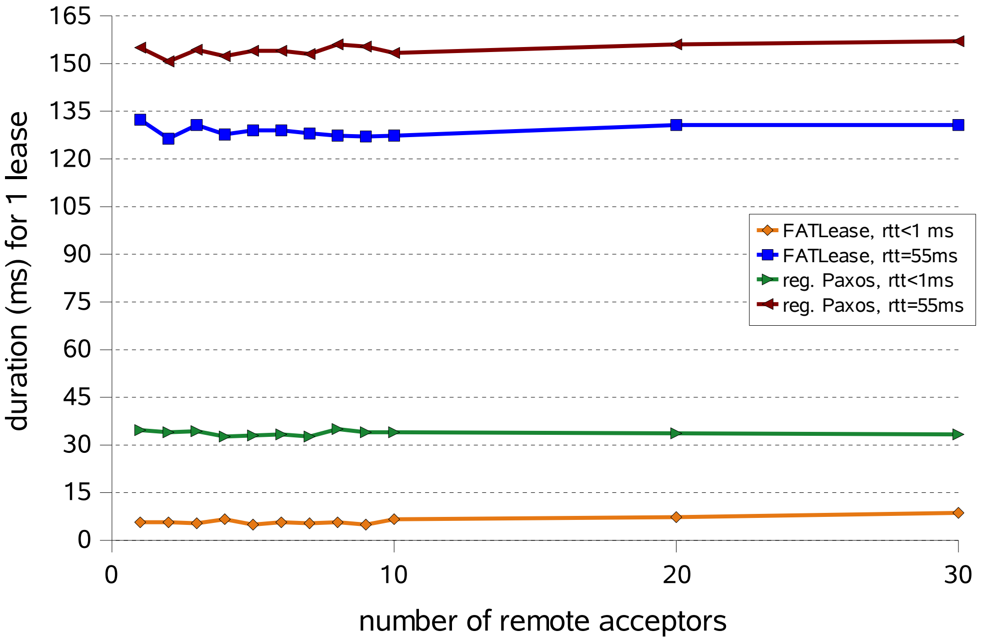
- Paxos needs stable storage to allow hosts to recover from a crash.
- With leases, each Paxos instance has a limited lifetime. After the lease has expired, the instance can be disposed.
- FaTLease does not require stable storage
- When a host recovers, it waits for the duration of a lease. Then, any instance it has participated in has timed out.



Evaluation of Scalability

- Scalability in terms of throughput and number of hosts





Conclusion

- FaTLease is a scalable and fault-tolerant algorithm for distributed leases
  - removes the need for a centralized lock service
- FaTLease is not limited by disk bandwidth
  - better performance than systems based on plain Paxos
  - suitable for "disk-centric" applications
- FaTLease can be used to make master/slave replication fault-tolerant (master fail-over)
  - advantage of separating data replication from fail-over mechanism
  - better performance than other fault-tolerant replication scheme: quorums

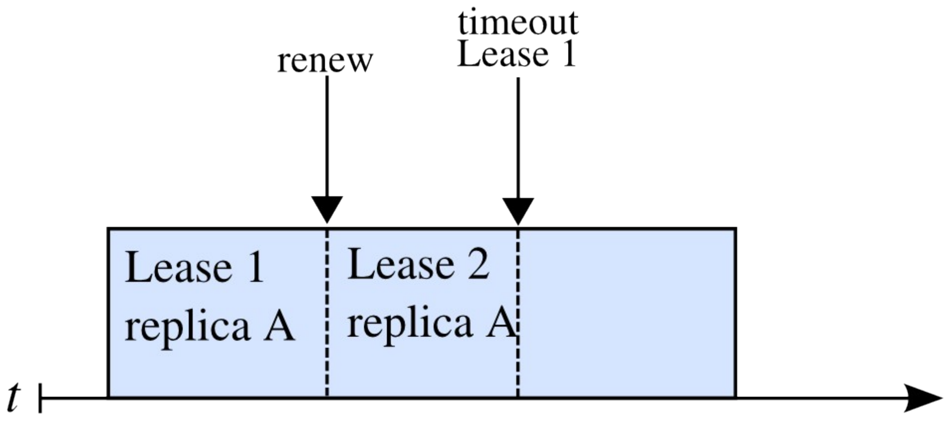
Thank you!

- References
  - Hupfeld et. al. "FaTLease: Scalable Fault-Tolerant Lease Negotiation with Paxos"
  - Burrows "The Chubby lock service for loosely-coupled distributed systems"
  - Lamport "Paxos Made Simple"

- Funded by the European Commision's FP6 programme under contract #FP6-033576

Renewing Leases

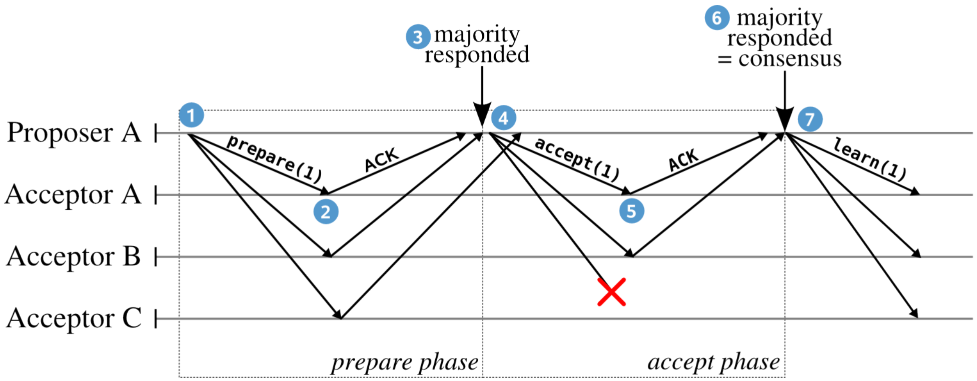
- Relax the "at most one valid lease at any time" to "all valid leases have the same lease owner"
- allow the lease owner to start a new instance before the current lease has timed out



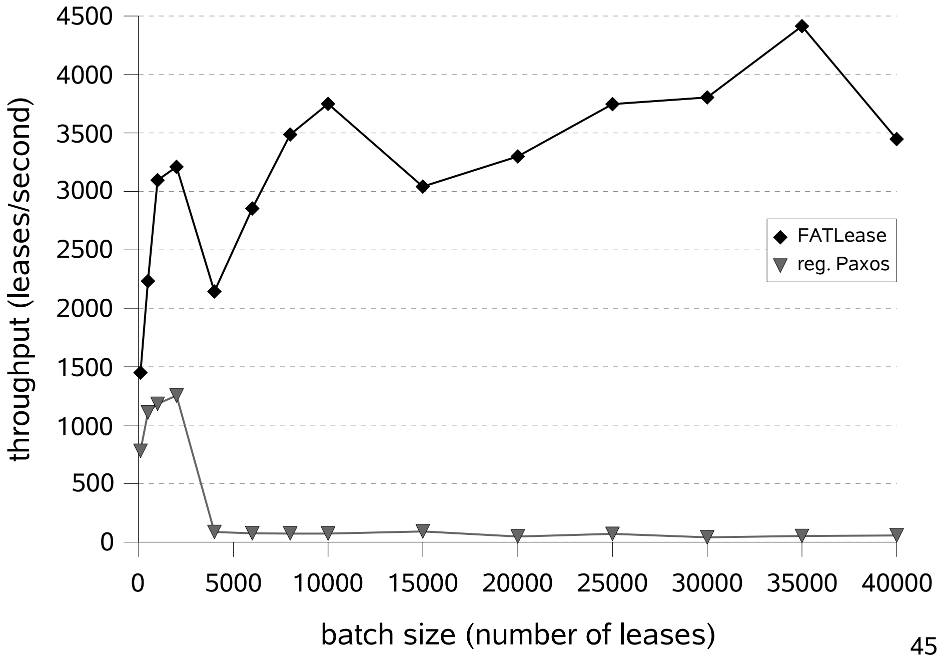
- FaTLease ensures that only the current owner can create a new instance

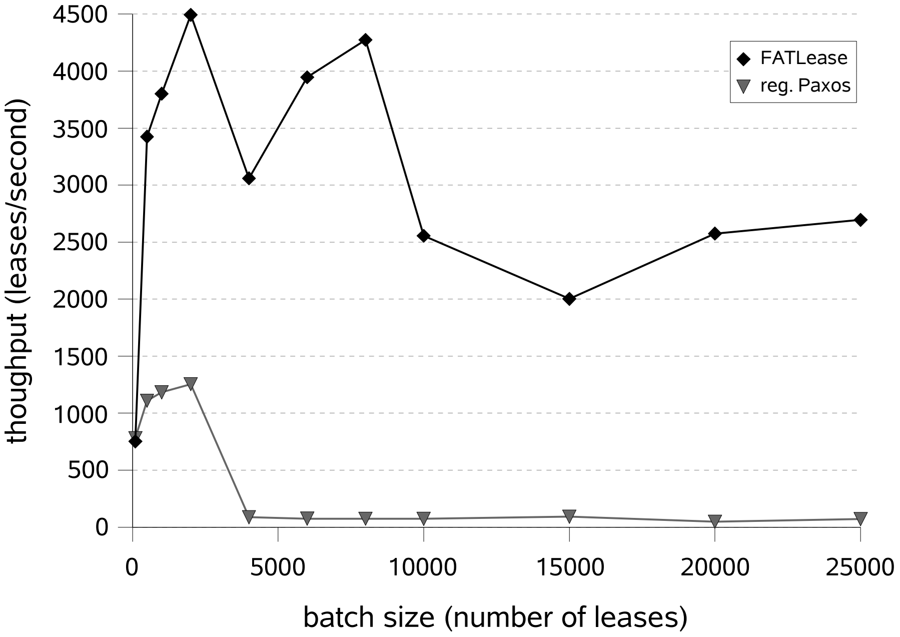
Paxos

- Relies on quorum decisions
  - can tolerate up to N/2-1 hosts to fail (e.g. message loss, crash, busy).



Scalability


throughput (leases/second)
batch size (number of leases)
◆ FATLease
▼ reg. Paxos


thoughput (leases/second)
batch size (number of leases)
◆ FATLease
▼ reg. Paxos