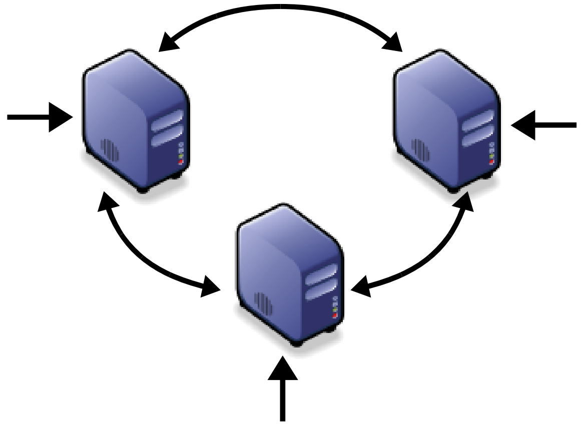
Flease - Lease Coordination Without a Lock Server

Björn Kolbeck, Mikael Högqvist, Jan Stender, Felix Hupfeld*
Zuse Institute Berlin, *Google Switzerland GmbH

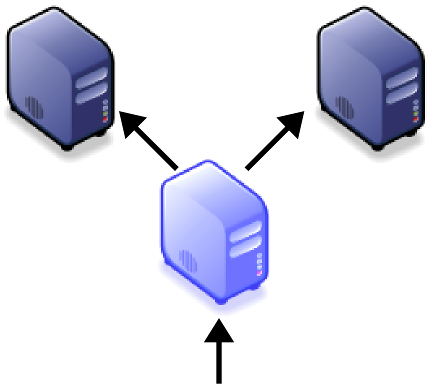
Problem: Data Replication

- Data replication with strong consistency
- Apply updates in same order
  ~ total order broadcast

Destination Agreement:
(Multi)Paxos



Fixed Sequencer:
Primary/Backup



Data Replication: Primary/Backup

- “Easy“ to implement
  - Single process takes all decisions
  - Widley used: Google GFS, many RDBMS (Oracle, DB2, MySQL)

- Primary is SPOF
  - Primary role must be revoked when process failed/disconnected

➔ Leases for Primary election
  - Lease: Exclusive access for limited period of time
  - Exclusive access = primary role
  - Timeout = revocation

Outline

1. Distributed Lease Coordination
2. The Flease Algorithm
3. Decentralized Lease Coordination
4. Evaluation

Distributed Lease Coordination

- Lease = exclusive access
- Lease Invariant:
  At most one valid lease at any point in time.

- Distributed System
  - Many processes concurrently trying to get a lease
  - All processes must agree on the same lease

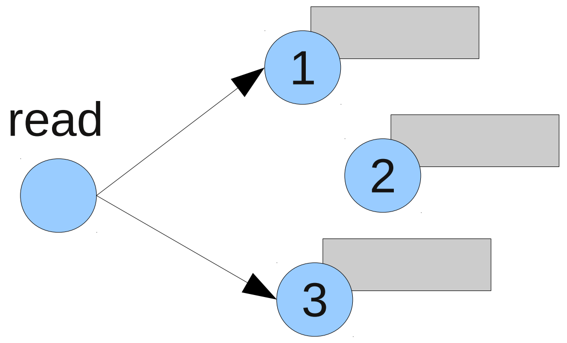
- Distributed Consensus (?)
  - (Multi)Paxos

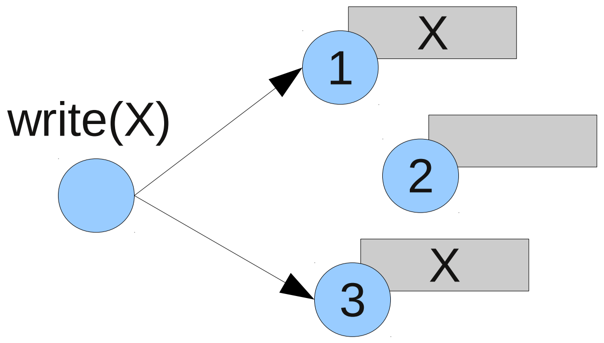
Distributed Lease Coordination: Agreement

- Agreement (Consensus):
  - If process p decides v then all process will decide v.
- Agreement (Leases):
  - If process p decides l then all process will decide l until l has timed out.
- Leases have a timeout.
  - We don't care about leases that have timed out

Deconstructing Paxos: Round Based Register

- Round-based register
  - Atomic read-modify-write
  - read(version)
  - write(version, new value)
- Register on each process
- Majority-based (Quorum Intersection Property)





Paxos vs. Flease

- Consensus with RBR

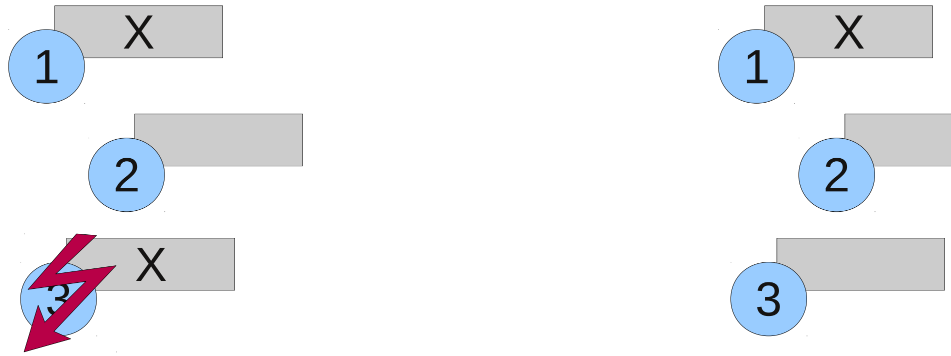
```
value = read(version)
IF value = empty THEN
    value := proposed value
END IF
IF write(value, version) THEN
    „decide“ value
END IF
```

- Lease Agreement with RBR

```
lease = read(version)
IF lease = empty OR timed_out(lease)
THEN
    lease := (me, $t_{now} + t_{max}$)
END IF
IF write(version, lease) THEN
    „decide“ lease
END IF
```

Flease: No persistent state

- Process crashes
  - Register contents is lost

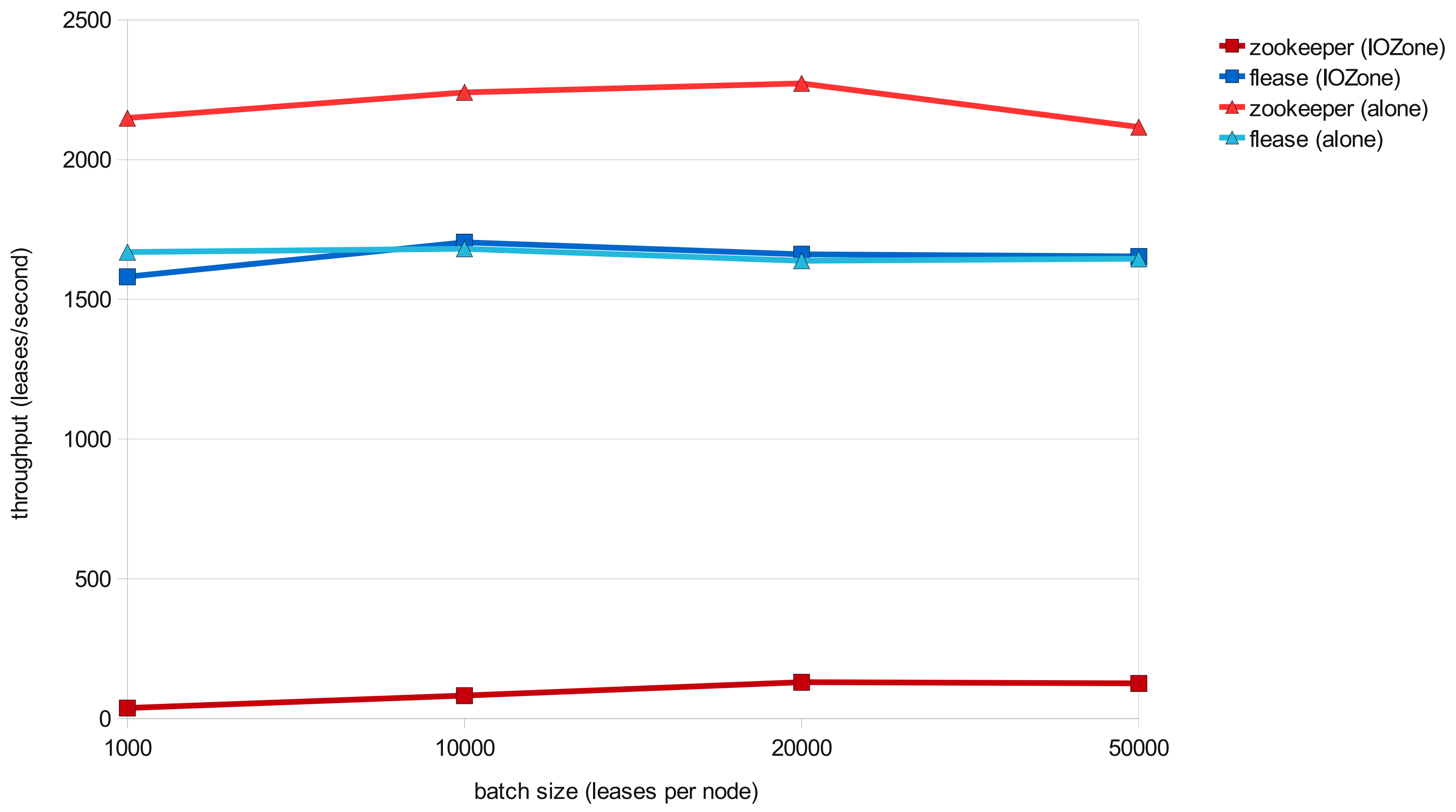


- Lease has timed out = empty register
  - `IF lease = empty OR timed_out(lease) THEN`
- Flease: wait for t_{max} before recovering
  - Lease in register has timed out

Advantages of Flease

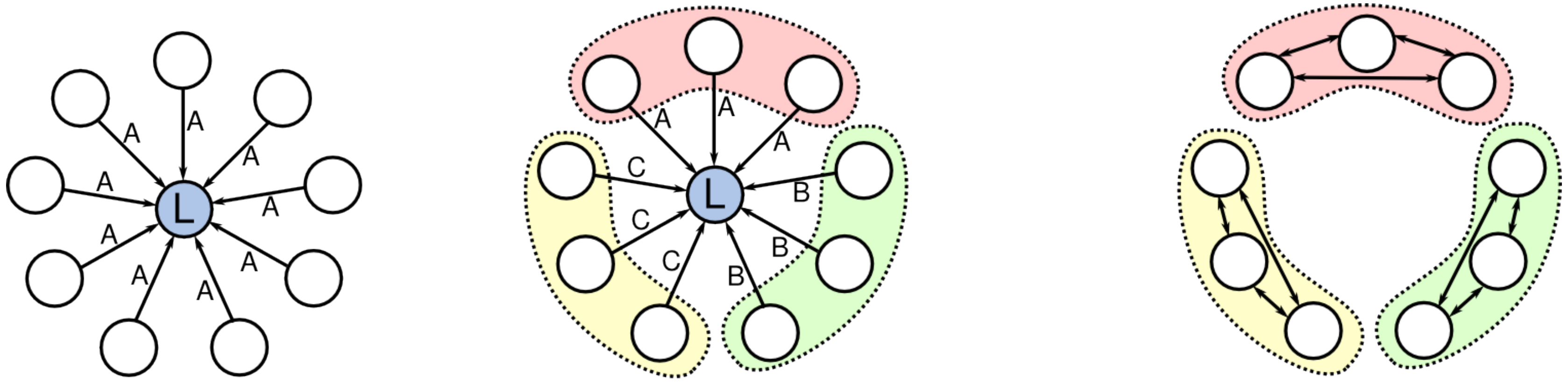
- Smaller state
  - Multipaxos: one Paxos instance per lease
  - Flease: only a single register
    - easier to implement
- No disk access
  - (Multi)Paxos: two writes per lease (on all nodes)
  - Flease: no disk writes
    - lower latency
    - throughtput limited only by bandwidth of RAM
    - share server with I/O intensive applications

Throughput under heavy IO load



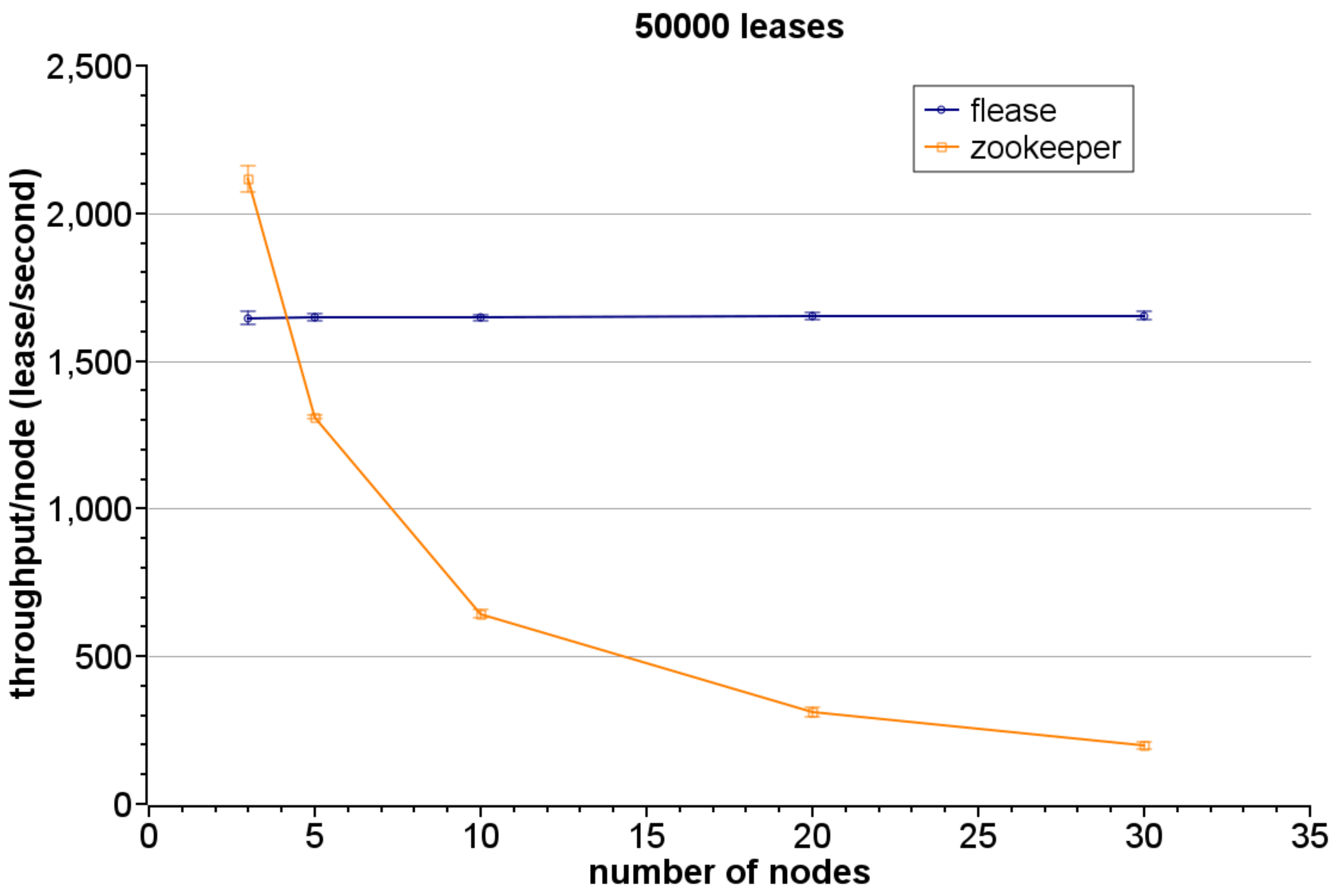
Decentralized Lease coordination

- No separate lock service



- Central Lock Service vs. Decentralized Leases
  - No extra service (saves hardware, maintenance)
  - Availability of replicas depends only on replica machines
  - Automatically scales with the system size

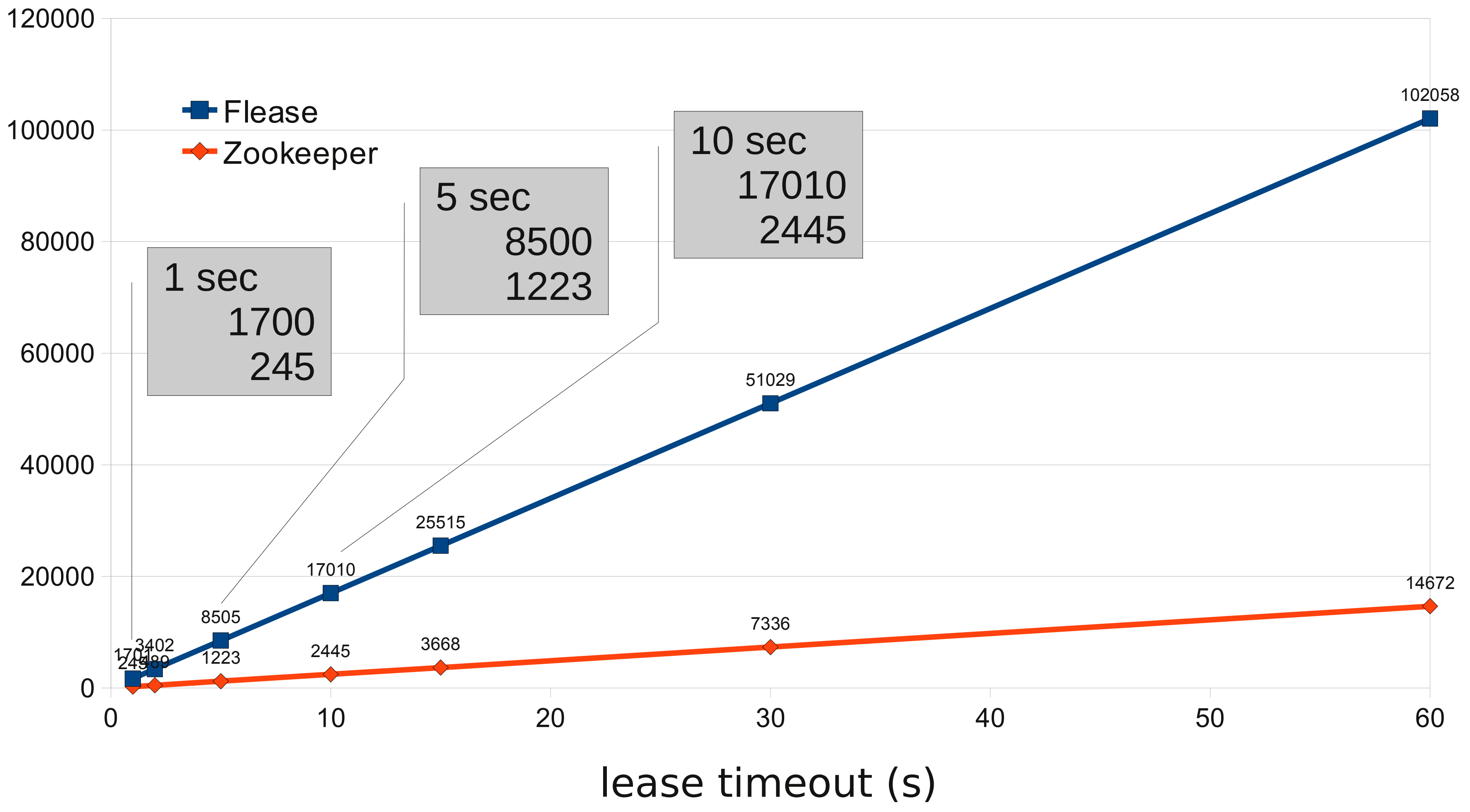
Evaluation: Scalability



– Zookeeper: 3 servers

– Flease: 3 nodes (2 randomly selected)

Evaluation: Max. number of open files/server



30 nodes, LAN

Thank You

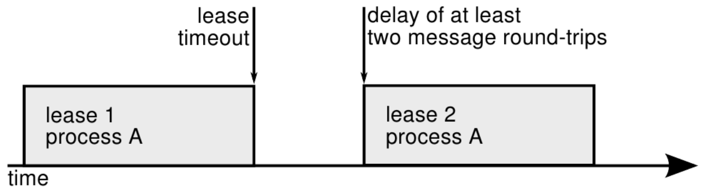
- Conclusion
  If you need a primary/exclusive access you can do better without a central lock service

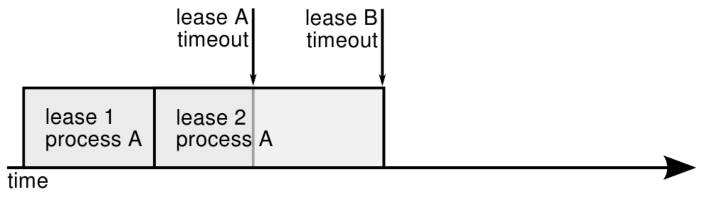
- Open Source implementation
  - www.xtreemfs.org

- www.contrail-project.eu

- The Contrail project is supported by funding under the Seventh Framework Programme of the European Commission: ICT, Internet of Services, Software and Virtualization.
  GA nr.: FP7-ICT-257438.

Flease: Renewing Leases





- Modified Lease Invariant:
  - If process p decides l=(p',t) then all process will decide l'=(p',t') with t' >= t until l has timed out.

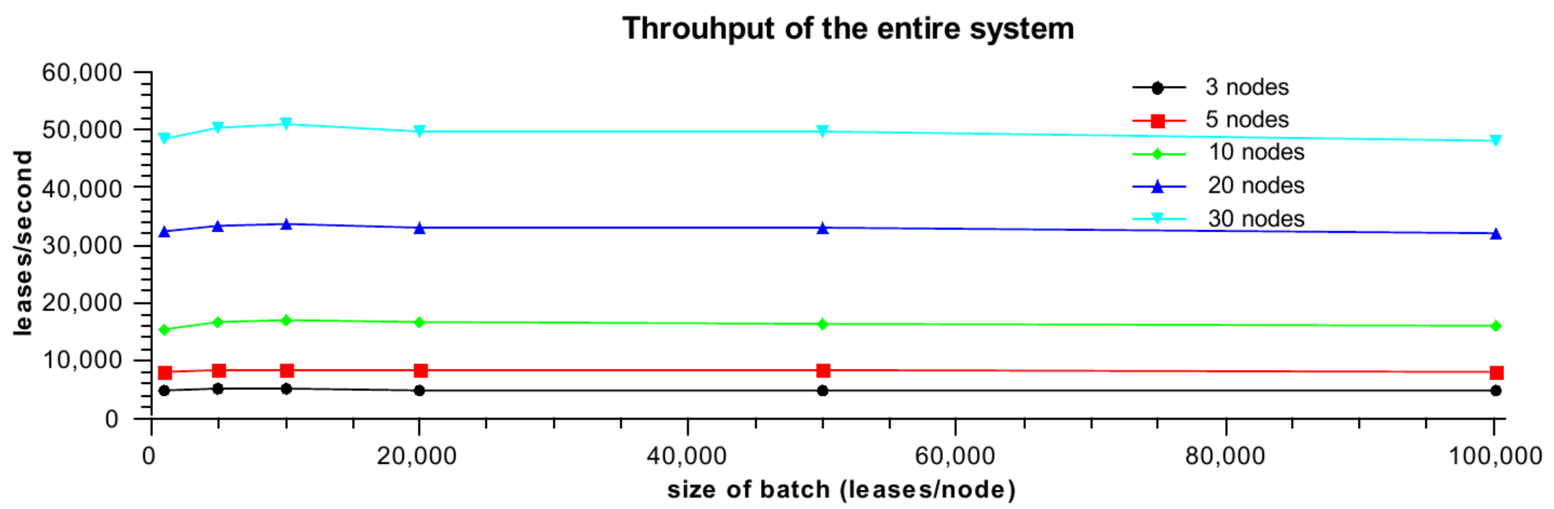
```
lease = **read**(version)
IF lease = empty
   OR timed_out(lease)
   OR owner(lease) = me THEN
    lease := (me, $t_{now}$ + $t_{max}$)
END IF
IF **write**(version, lease) THEN
    „decide“ lease
END IF
```

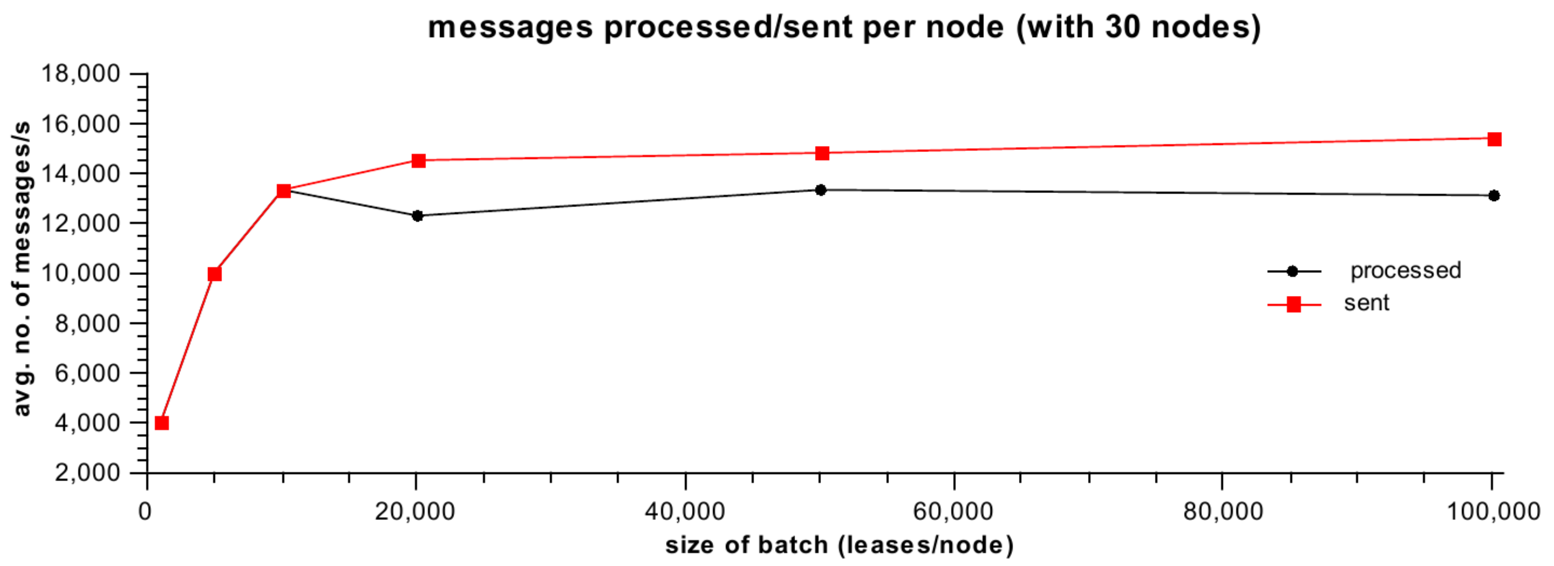
Flease: The other half of the truth.

- Assumed perfectly synchronized clocks
- Instead: Loosely synchronized clocks
  - $c(t) < c(t')$ if $t < t'$
  - At any time t for any two processes p, q: $| c_p(t) - c_q(t) | < \varepsilon$
  - ε system-wide constant, e.g. 1 sec

```
lease = read(version)
IF lease.t < $t_{now}$
   AND lease.t > $t_{now}$ + ϵ THEN
   wait ϵ
   retry
END IF
...
```

Throughput vs. Messages





XtreemFS: Flease for file replication

- One lease per file = one primary per file
  - better load balancing
  - arbitrary replica placement
- When a file is openend
  - Elect a primary with Flease
  - Execute Replica Reset
  - Read locally, write quorum