Universit□t
Salzburg

VO Distributed Systems
Data Replication & Resilience

Dipl.-Ing. Michael Holzmann
Software Research Lab

www.SoftwareResearch.net

SOFTWARE RESEARCH LAB

Overview

- Atomic actions (transactions) in distributed systems
- Make transactions resilient: data replication
- Introduction replica management
- Replica management strategies:
  - Version vectors
  - Primary site
  - State machine (active replication)
  - Voting
    - majority
    - hierarchical
- Degree of replication

Atomic Actions

- Atomic action: A couple of identified operations appear as atomic.
- E.g.: Money transfer

```
begin T1                 begin T2
read account A           read account A
read account B           read account B
add €500 to B            add €200 to B
sub €500 from A          sub €200 from A
update A                 update A
update B                 update B
end T1                   end T2
```

Atomic actions in distributed systems

- Either completes successfully or it appears as if the action has not executed at all.
- How is this done?
- What can prevent atomic actions from successful completion (violation of atomicity)?
- Data access: uniprocess environment vs distributed system
- What happens in a distributed system where multiple copies of data exist in the case of failures?

Transactions: completion vs rollback

- Applications where rollback makes no sense?
- If we are interested rather in successful completion than in roll-back:
  - How can this be achieved? / Problems to be dealt with?

- Access Control: no concurrent data access
- How can we complete actions even in the case of failures?
- Replicate data items needed during the transaction

Data replication

- Purpose of replication: support fault tolerance -> replication should not be visible.
- Operations on logical data items are mapped to operations on the multiple copies of the data items.
- One-copy serializability criterion
  - Has to be guaranteed by replica control algorithms
- Which types of failures have to be handled by replica control algorithms in a distributed system?
  - Node failures
  - Communication failures

Dealing with communication failures

- How do node failures affect the system behavior?
  - As long as one copy of the data item is accessible the operation can be finished.
- What can happen in the case of communication failures?
  - Network partitioning: Nodes in the different partitiones are unable to communicate with each other.
  - Whitin a group the consistency of the replicas can be preserved, but
  - Global mutual consistency of the replicas may be violated.

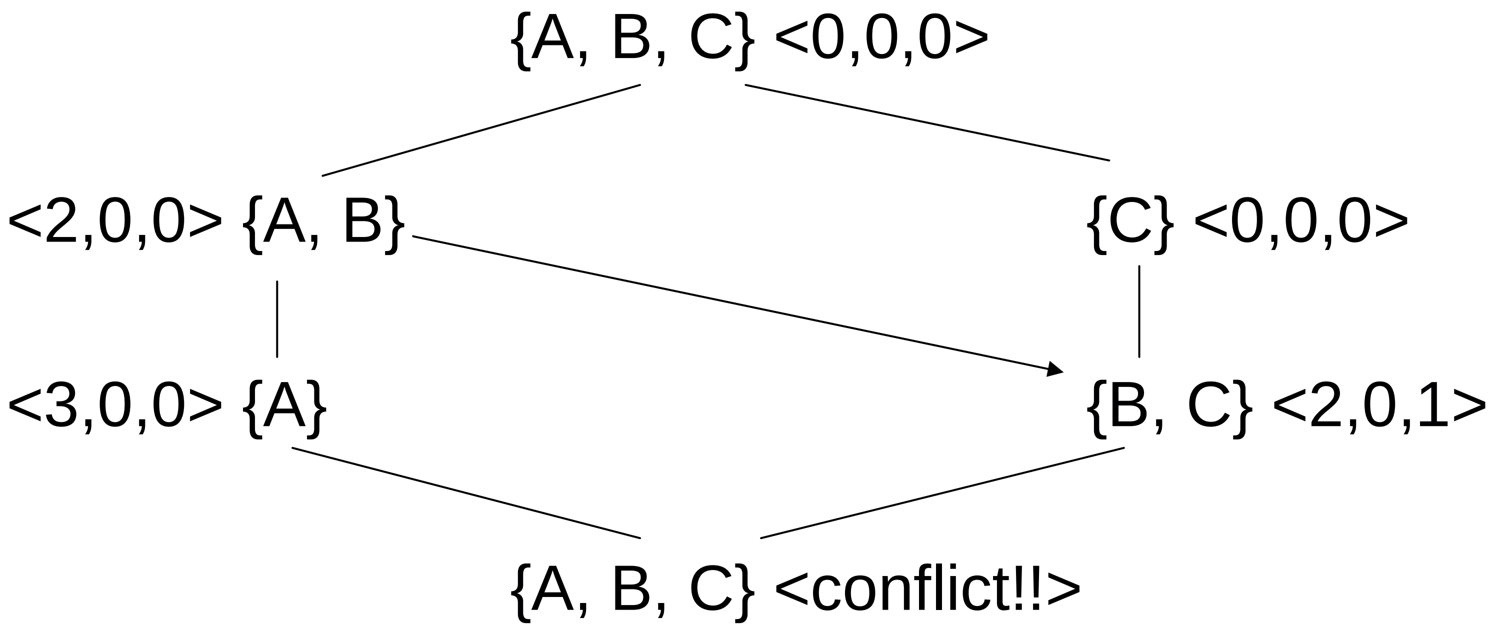
Replica control methods

- To avoid inconsistencies in the case of network partitioning processing in the different partitions has to be restricted!
- Two different approaches:
  - Optimistic: no restrictions, solve inconsistencies after rejoin (hopefully).
  - Pessimistic: restrictions on processing.

Optimistic approaches

- Operations are performed independently in each group.
- Serilizability in each group can be preserved, but global inconsistencies may arise: solve them after rejoin.
- Version Vectors:
  - Each replica has a version vector V[n] which reflects the number of updates of the copy by the different nodes.
  - Relationship between two vectors:
    - Dominate: V[i] > V[i]‘ for all i = 1..n
    - Conflict: neither vector dominates the other
  - If two copies conflict, it is up to the system manager to manually do what ever necessary.

Version Vectors Example



- Only detection of inconsistencies due to update
- No detection of read-write conflicts

Primary Site Approach

- Pessimistic Approach
- Handles node failures.
- The system must be able to distinguish node and communication failures in order to handle partitioning.
- To support *k-resilient* data, the data is repicated on k+1 nodes
- Most straightforward approach: designated node is ‚primary', it coordinates update of the ‚backup' nodes.

Primary Site Approach – handling failures

- What happens in the case of failures:
  - Backup node:
    - service is not disrupted
  - Primary node:
    - new primary has to be elected. Continue user requests after finishing actual request / continue from checkpoint.
  - Network partitioning:
    - The partition with the primary node continues servicing requests. The others can not continue.
- Advantage of primary site approach regarding update policy of the backup nodes:
    - Any centralized concurrency control protocol can be used.

State Machine Approach

- All replicas are simultaneously active.
- Can handle only node failures, but is able to handle even Byzantine failures.
- Main idea: ensure that all replicas get the same sequence of requests. Prerequisites:
  - Agreement: all replicas receive every request
  - Order: all replicas process requests in the same order
- How to achive these requirements?
  - Byzantine agreement protocol / reliable broadcast
  - Assigning unique IDs: logical clock algorithm.
  - Satisfies both requirements: Atomic Broadcast.

State Machine using Atomic Broadcast

- Example: Implementing Resilient Objects
- Problem: A requested operation on O1 may request an operation on another object O2. What will happen?
  - All nodes performing operation on O1 will also request the operation on O2 -> *images*. Consistency?
- Each *independent* operation is assigned a unique ID
- All *images* of an independent request have the same ID
- Realization:
  - Each node has a counter that is incremented whenever message is received or broadcasted.
  - Top-level: OpID = node# + counter
  - Nested-level: OpIDn = OpID + sequence#

State Machine using Atomic Broadcast

- Each operation is performed once at each replica:
- Each node maintains a request and a result queue.
  - Requests / Results are only broadcasted if there is no copy of this request / result in the queue.
- After a request has been serviced, what happens to its copy in the request queue?
    - It has to be kept until no other copy of the request can arrive.
    - Purged after ‚sufficient' time has elapsed (commu. delays)
    - If a request is served, the copy in the queue is marked.
  - ‚Marking' means, that the operation has been performed on the local replica.

State Machine – atomic actions

- So far we have focussed on single operations.
- How to preserve One-copy serializability?
  - Requirement: all replicas use the same concurrency control method.
    - Atomic broadcast ensures that all replicas receive the requests in the same order and thus execute the same steps.
- Example: 2-phase locking protocol
- Reintegration of failed nodes:
  - State transition from other nodes

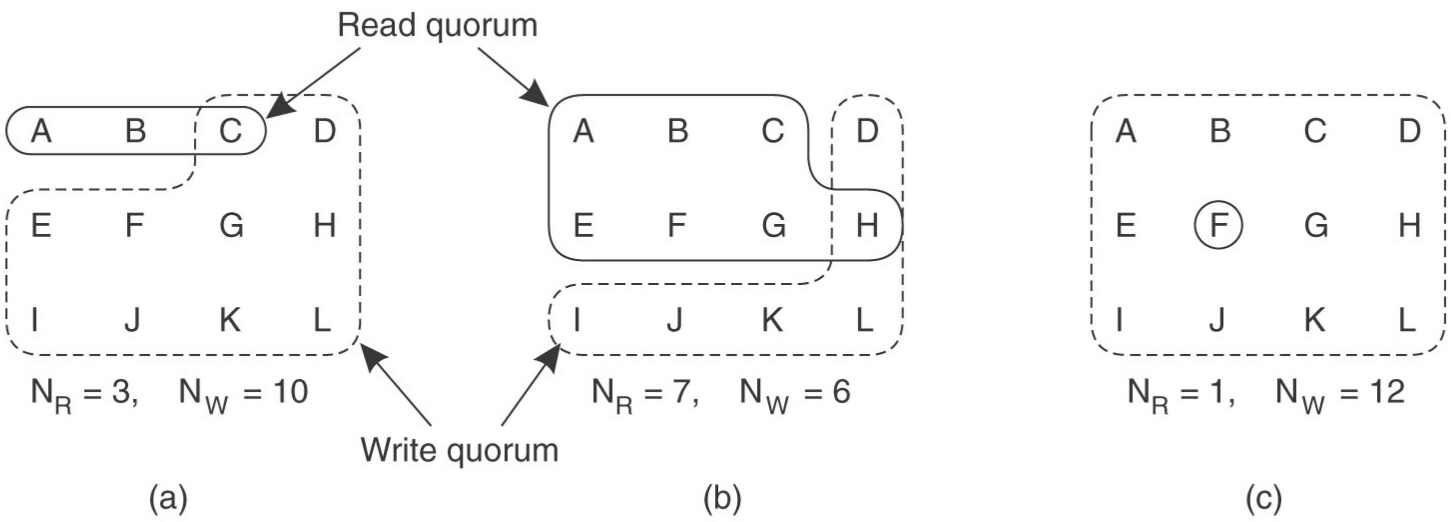
Voting

- Performing actions on replicas is decided collectively by voting: conflicting operations not performed concurrently.
- Advantage: masks both, node and communication failures
- Voting Methods:
  - Static: Assignment of votes is predefined statically
  - Dynamic: Assingment of votes may change to adapt to changings of the system state.

Weighted Voting

- Each node is assigned some number of votes
- Any node that wants to read / write data has to aquire at least r / w votes from the system before it may proceed.
  - r + w > v , v ... sum of all votes.
  - w > v / 2
- What do these conditions guarantee?
  - Every read and write quorum intersect
  - Every read quorum contains one replica with the latest update
  - Two write quorums intersect
  - In the case of partitioning write is allowed to be performed in at most one group

Weighted Voting: Quorums



- Extremes:
  - r=1, w=v (ROWA). No updates possible if single node fails
  - r = w = v/2+1. If partitioning occurs only majority group functions

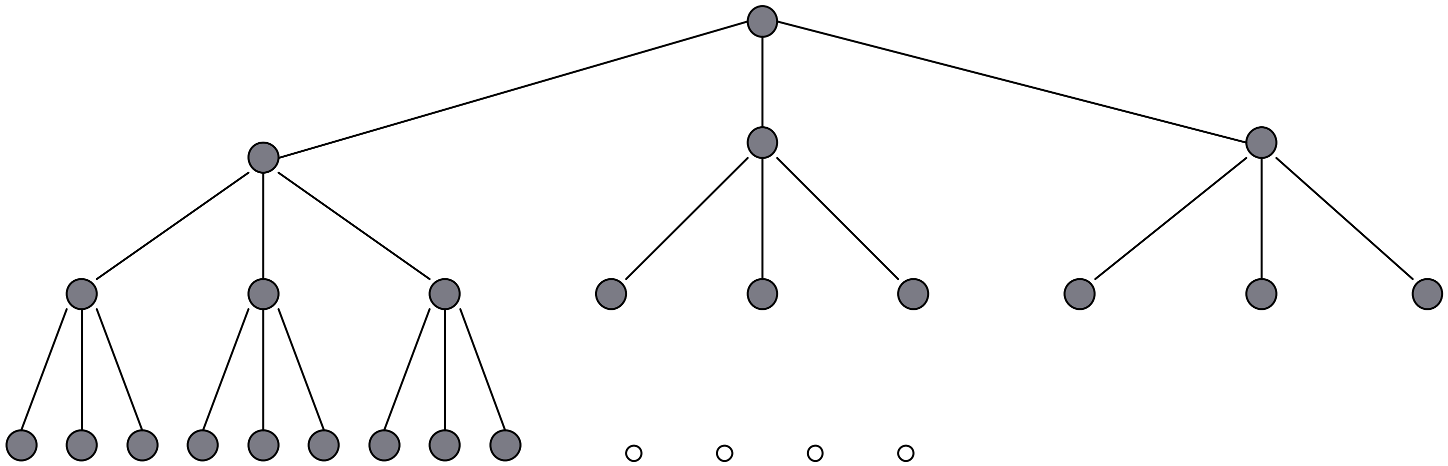
Weighted Voting - operations

- Each node broadcasts a request for votes.
- All nodes reply with the number of votes they posses and their *version number* of the replica.
- If the requesting node has aquired sufficient votes (>=quorum) it starts to perform the operation.
  - Read operation: check the received version numbers, at least one replica has the latest update.
  - Write operation: the requester makes sure that all nodes in the quorum are written using the latest value.

Hierarchical Voting

- Problem with majority voting:
  - # of nodes required in a quorum increases linearly with # of replicas.
- Hierarchical approach: reduce number of nodes that must be in a quorum by introducing a multi-level structure (tree): each level corresponds a quorum.
- Aquiring a quorum at level i implies quorum collection right down to the leaf level.
- Conditions for the quorum at each level i:
  - $r_i + w_i > l_i$ (l_i .. # of childreen at level i)
  - $2w_i > l_i$

Hierarchical voting - quorums



- What is the condition for the quorums in this case?
- ri + wi > 3, wi >= 2
- r/w quorums which result in fewest nodes / quorum?
- ri=2 and wi=2, i=1..3 -> only 8 nodes are in the quorums!

Degree of Replication

- Increasing degree of replication:
  - Increased reliability -> increases availability
  - Increased overhead -> decreases availability
- What is the optimum degree of replication?
- Example: Primary Site Approach
  - The state of the primary is periodically checkpointed on all backups at a certain frequency f.
  - If the primary fails, all operations from the checkpoint are redone by the new primary (recovery).
  - Failed sites are repaired and rejoin after repair.

Optimal Degree of Replication

- Availability A is the fraction of time the system is availiable for user requests:
  - A = 1 – O/L ; L .. period of observation.
- Overhead O occurs because of checkpointing, recovery and repair:
  - A = (1 – Ti)(1 – Tr)(1 – Tc)
  - Increasing N reduces Ti
  - But increasing N increases Tc at constant f.
- Practical examples have shown that availabilities with N=2 or N=3 are only 1% less optimal availability.

Universit□t Salzburg

VO Verteilte Systeme

Thank you for your attention!

SOFTWARE RESEARCH LAB