Data replication, consistency models & protocols

C. L. Roncancio - S. Drapeau
Grenoble INP – Ensimag / LIG - Obeo

Replication

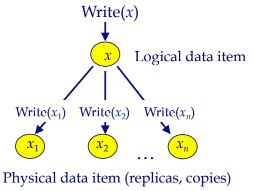
- ✓ Data and process
- ✓ Focus on data: several physical copies of one logical object

What for?

- ✓ Performances, availability, reliability
- ✓ Scalability
- ✓ Application requirements

Execution Model

- ✓ There are physical copies of logical objects in the system.
- ✓ Operations are specified on logical objects, but translated to operate on physical objects.
- ✓ One-copy equivalence
  - – The effect of transactions performed by clients on replicated objects should be the same as if they had been performed on a single set of objects.



©M. T. Özsu & P. Valduriez

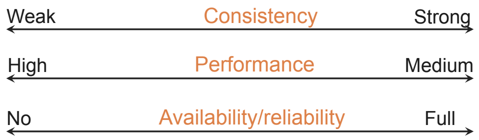
Difficulties

- ✓ Transparency of replication
- ✓ Global coherency
  - – *Mutual consistency* among copies
  - – Transactional coherency
- ✓ Mutual consistency
  - – Cost
  - – Impact on the availability

Widely used

- Distributed systems
- Parallelism: fragmentation (*sharding*) & replication
- Contexts with few conflicting updates
  - Data warehouses, read only systems..

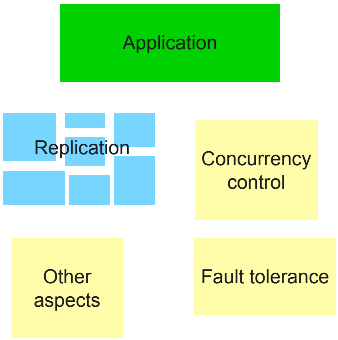
Mutual consistency, what for?



Many protocols with different

- performances
- fault tolerance guaranties
- transparency
- Programming difficulties
- …

Several aspects



Application

Replication

Concurrency control

Other aspects

Fault tolerance

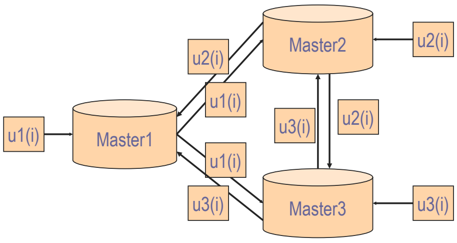
Notions from distributed systems

- ✓ Consistency model
  - Consistency perceived by the programmers/users
- ✓ Consistency protocol
  - How the system insures de consistency model
  - Degree of synchronization of the copies
  - Update propagation, invalidation

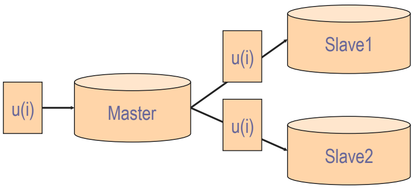
Types of replication

- ✓ Symmetric
  - Updates on any copy
- ✓ Asymmetric
  - Updates on some selected copies
  - Other copies are read-only
- ✓ Snapshots
  - Materialized views

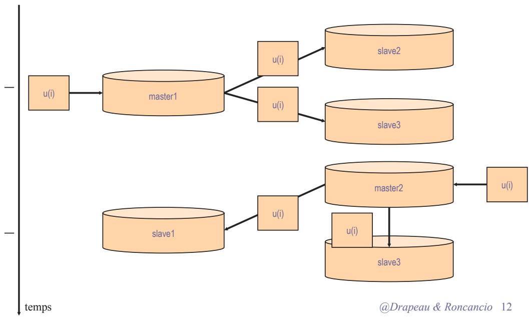
Symmetric replication



Asymmetric replication (1)



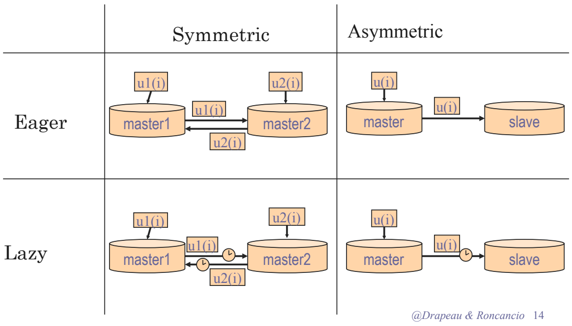
Asymmetric replication (2)



Update propagation (1)

- ✓ *Eager*
  - – Update of the copies in a single transaction
    - Synchronous
- ✓ *Lazy*
  - – Update of the copies in separate transactions
    - Asynchronous

Update propagation (2)



One-copy serializability (1SR)

The execution of the transactions is equivalent to a sequential execution on non replicated data

- ✓ Eager and lazy strategies

Mutual consistency

- ✓ Strong consistency
  - – When reading any copy the user gets a value including all the preceding updates
- ✓ Weak consistency
  - – Copies may not have the “last” value but all updates will be eventually propagated to all the copies
  - – **Eventual consistency:** copies will converge to a single value in a finite time

Update propagation strategies

Eager

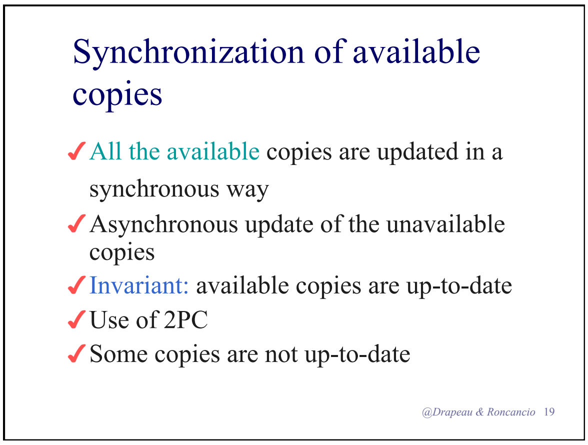
- ✓ Full synchronization
- ✓ Synchronization of available copies
- ✓ Quorum

Lazy

- ✓ Master / slaves
- ✓ Master / secondary
- ✓ Lazy asymmetric (independent)

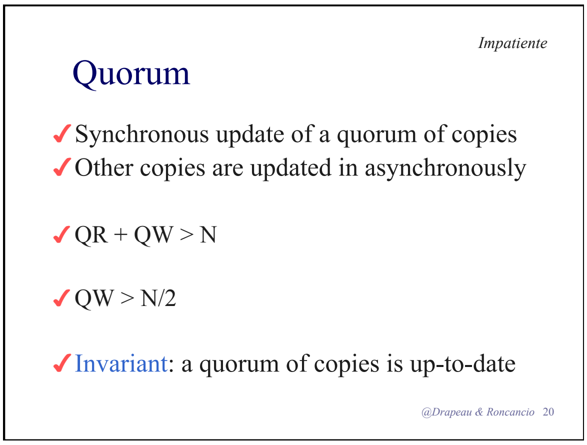
Full synchronization

- ✓ All the copies are updated in a synchronous way
- ✓ Atomic update of the copies
  - – 2PC, Concurrency control requirements



Synchronization of available copies

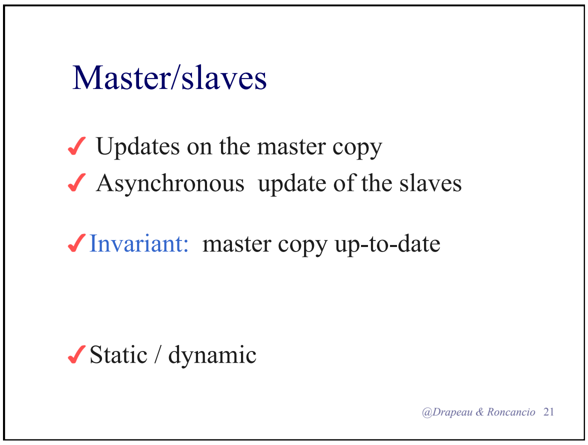
- ✓ All the available copies are updated in a synchronous way
- ✓ Asynchronous update of the unavailable copies
- ✓ Invariant: available copies are up-to-date
- ✓ Use of 2PC
- ✓ Some copies are not up-to-date



Impatiente

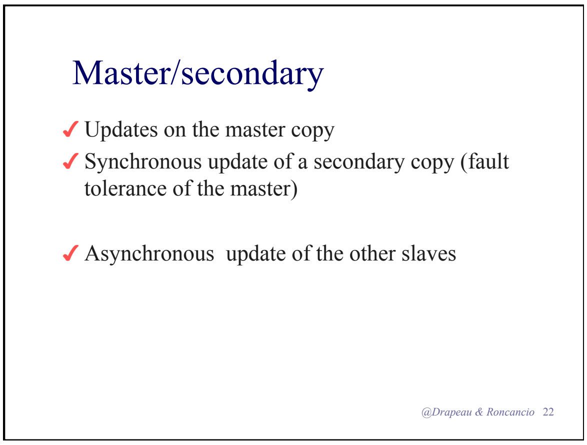
Quorum

- ✓ Synchronous update of a quorum of copies
- ✓ Other copies are updated in asynchronously
- ✓ QR + QW > N
- ✓ QW > N/2
- ✓ Invariant: a quorum of copies is up-to-date



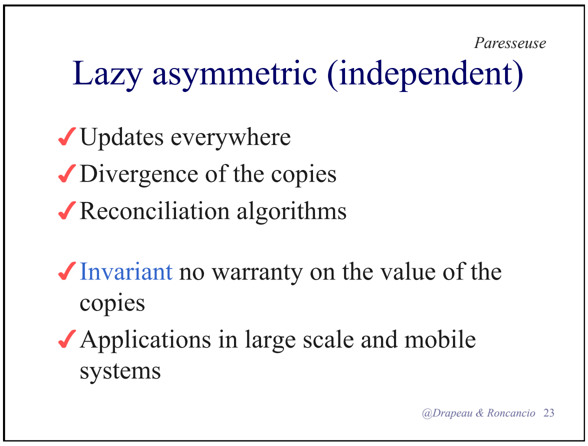
Master/slaves

- ✓ Updates on the master copy
- ✓ Asynchronous update of the slaves
- ✓ Invariant: master copy up-to-date
- ✓ Static / dynamic



Master/secondary

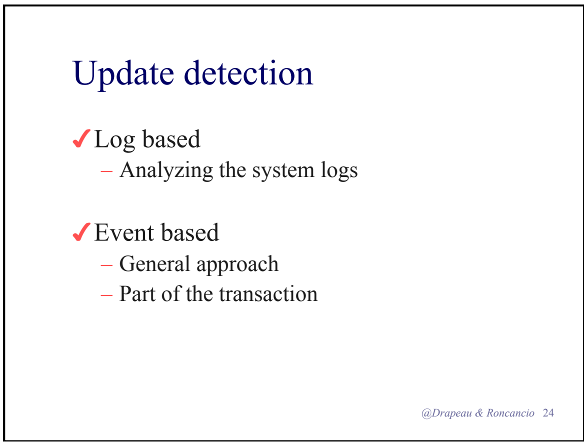
- ✓ Updates on the master copy
- ✓ Synchronous update of a secondary copy (fault tolerance of the master)
- ✓ Asynchronous update of the other slaves



Paresseuse

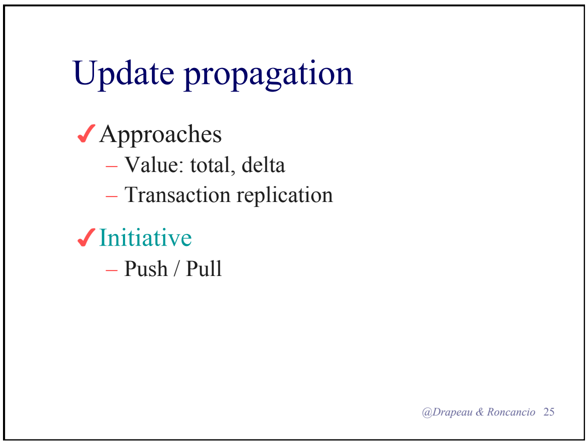
Lazy asymmetric (independent)

- ✓ Updates everywhere
- ✓ Divergence of the copies
- ✓ Reconciliation algorithms
- ✓ Invariant no warranty on the value of the copies
- ✓ Applications in large scale and mobile systems



Update detection

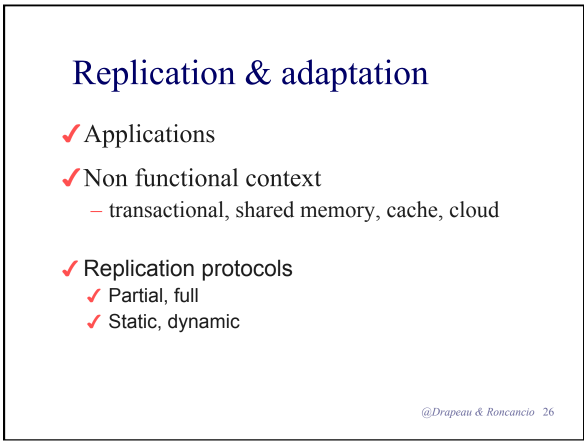
- ✓ Log based
  - Analyzing the system logs
- ✓ Event based
  - General approach
  - Part of the transaction



Update propagation

✓ Approaches
  - Value: total, delta
  - Transaction replication

✓ Initiative
  - Push / Pull

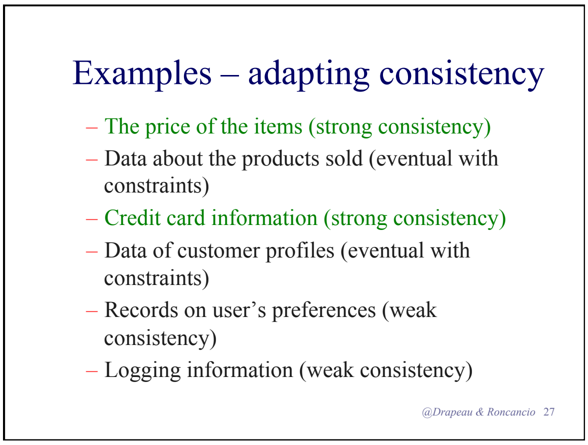


Replication & adaptation

✓ Applications

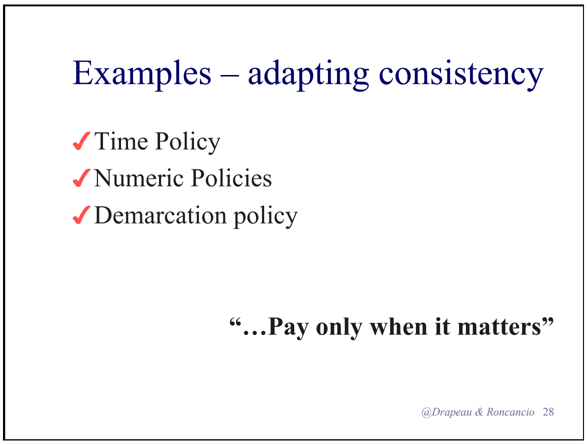
✓ Non functional context
  - transactional, shared memory, cache, cloud

✓ Replication protocols
  - ✓ Partial, full
  - ✓ Static, dynamic



Examples – adapting consistency

- The price of the items (strong consistency)
- Data about the products sold (eventual with constraints)
- Credit card information (strong consistency)
- Data of customer profiles (eventual with constraints)
- Records on user’s preferences (weak consistency)
- Logging information (weak consistency)



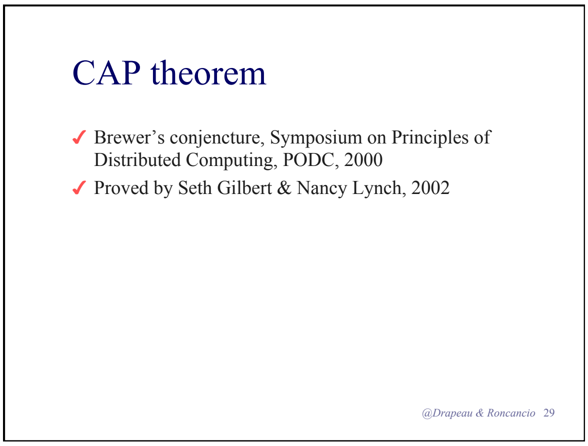
Examples – adapting consistency

✓ Time Policy

✓ Numeric Policies

✓ Demarcation policy

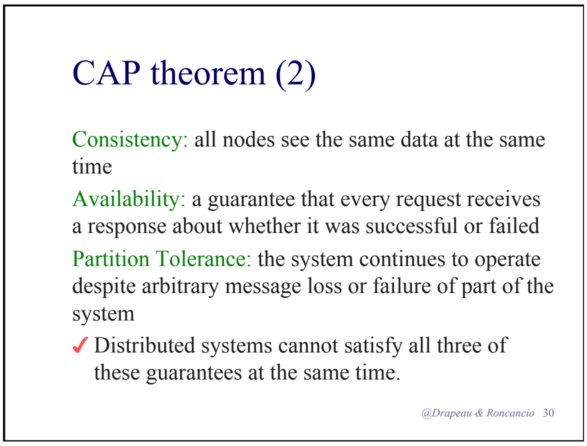
“…Pay only when it matters”



CAP theorem

✓ Brewer’s conjecture, Symposium on Principles of Distributed Computing, PODC, 2000

✓ Proved by Seth Gilbert & Nancy Lynch, 2002



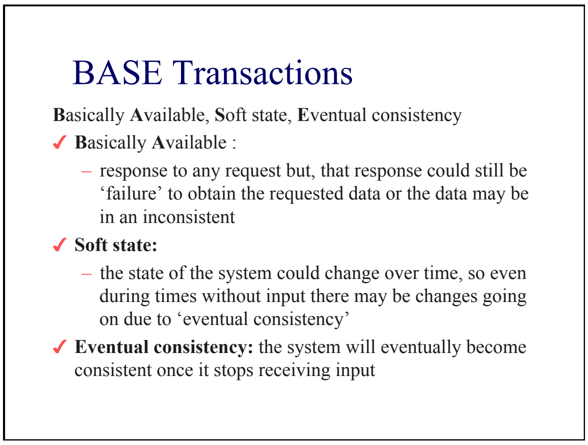
CAP theorem (2)

Consistency: all nodes see the same data at the same time

Availability: a guarantee that every request receives a response about whether it was successful or failed

Partition Tolerance: the system continues to operate despite arbitrary message loss or failure of part of the system

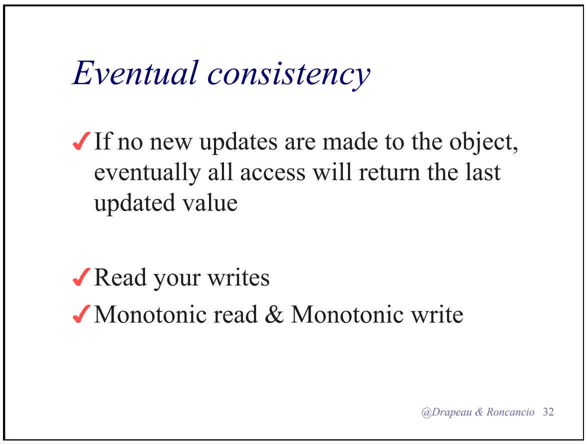
✓ Distributed systems cannot satisfy all three of these guarantees at the same time.



BASE Transactions

Basically **A**vailable, **S**oft state, **E**ventual consistency

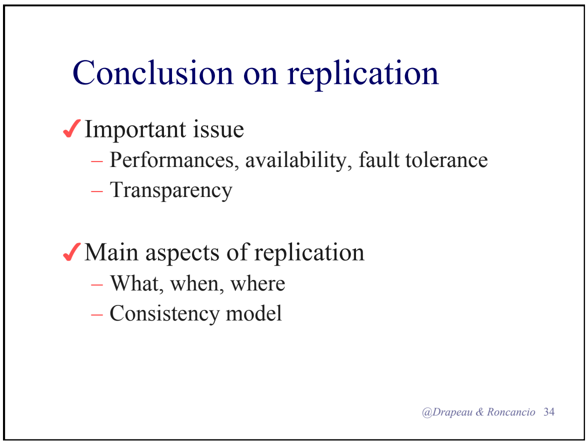
- ✓ **B**asically **A**vailable :
  - response to any request but, that response could still be ‘failure’ to obtain the requested data or the data may be in an inconsistent
- ✓ **Soft state:**
  - the state of the system could change over time, so even during times without input there may be changes going on due to ‘eventual consistency’
- ✓ **Eventual consistency:** the system will eventually become consistent once it stops receiving input



Eventual consistency

- ✓ If no new updates are made to the object, eventually all access will return the last updated value

- ✓ Read your writes
- ✓ Monotonic read & Monotonic write



Conclusion on replication

- ✓ Important issue
  - Performances, availability, fault tolerance
  - Transparency

- ✓ Main aspects of replication
  - What, when, where
  - Consistency model