Consistency & Replication in Distributed Systems

Distributed Systems

Sistemi Distribuiti

Andrea Omicini

andrea.omicini@unibo.it

Dipartimento di Informatica – Scienza e Ingegneria (DISI)

Alma Mater Studiorum – Università di Bologna a Cesena

Academic Year 2013/2014

Outline

These Slides Contain Material from [Tanenbaum and van Steen, 2007]

Slides were made kindly available by the authors of the book

- Such slides shortly introduced the topics developed in the book [Tanenbaum and van Steen, 2007] adopted here as the main book of the course
- Some of the material from those slides has been re-used in the following, and integrated with new material according to the personal view of the teacher of this course
- Every problem or mistake contained in these slides, however, should be attributed to the sole responsibility of the teacher of this course

Outline

Reasons for Replication

Replication of Data

- Increasing the reliability of systems
- Improving performance
- Scaling
  - in numbers
  - in geographical area

Issues of Replication

Benefits in Distributed Systems

- Reliability
- Fault tolerance
- Accessibility
- Performance
- Scalability

Issues of Replication

Benefits in Distributed Systems

- Reliability
- Fault tolerance
- Accessibility
- Performance
- Scalability

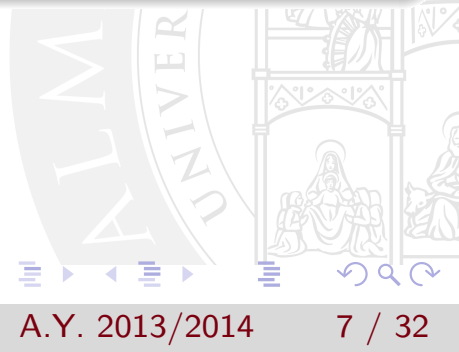
Problems in Distributed Systems

- Costs
  - computational resources
  - bandwidth
- Consistency

The Problem of Consistency

Consistency models

- Before discussing how to face the problem of consistency, we need to define the notion of consistency itself
- Different interpretations are available, some of them fitting one or more application scenarios
- This leads to the definition of different *models of consistency*...
- ... which are then amenable of different implementations, whose features may affect the effectiveness of the model



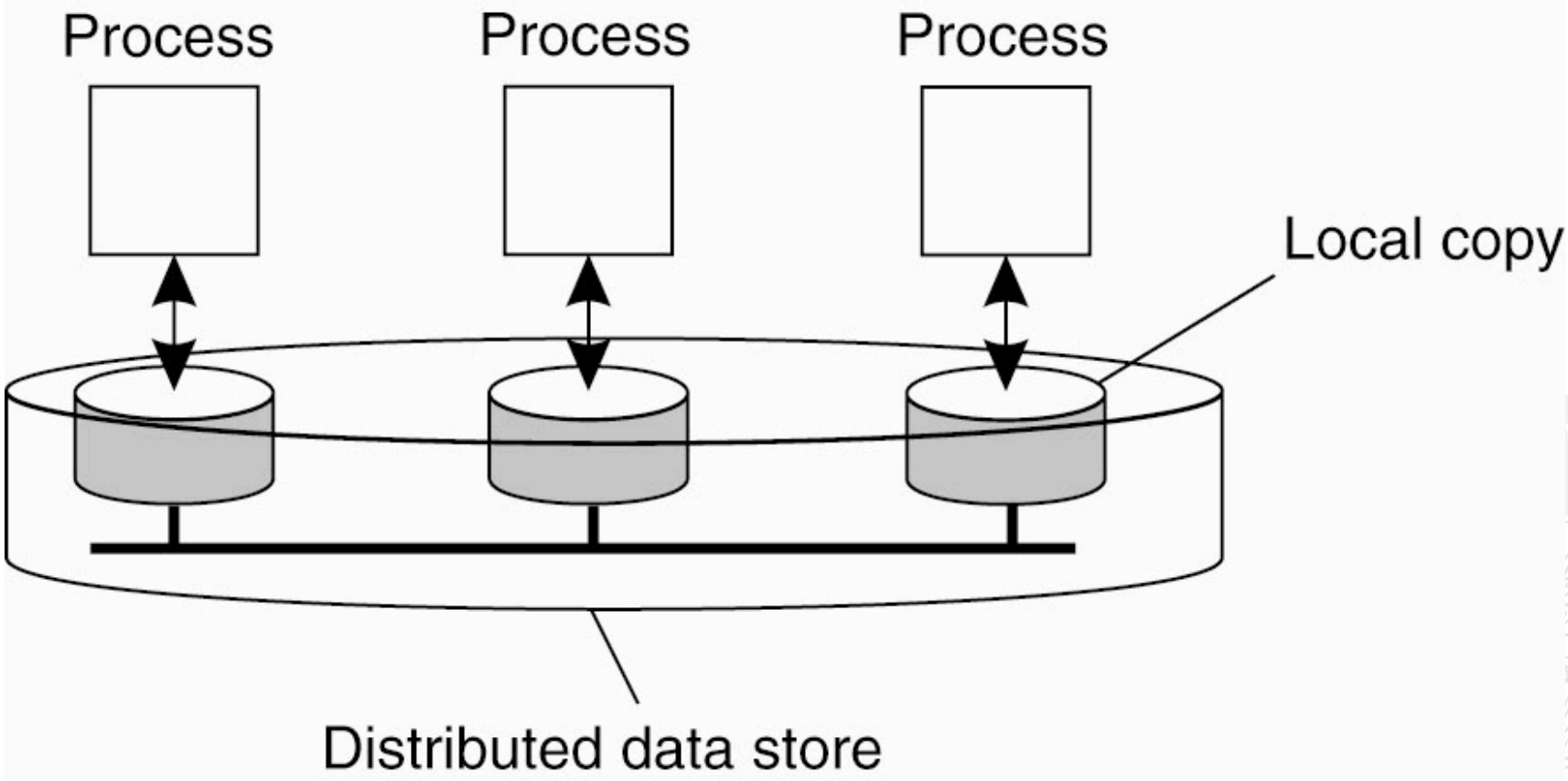
Outline

Conceptual Premise

Replicating data

- Historically, the first things to be distributed are data
- So, the first problem to be addressed is how to ensure *consistency of data* across distributed copies...
- ... and the actions to be accounted for are *operations over data*

General Organisation of a Distributed Data Store



[Tanenbaum and van Steen, 2007]

Consistency Model

Definition

A *consistency model* is essentially a contract among the processes and the data stores, ensuring the correctness of data given a set of rules that processes have to follow

Of course, what is "correct" also depend on what processes expect – which might be also difficult to define in absence of a global clock

Consistency Model

Definition

A *consistency model* is essentially a contract among the processes and the data stores, ensuring the correctness of data given a set of rules that processes have to follow

Of course, what is "correct" also depend on what processes expect – which might be also difficult to define in absence of a global clock

Observation

The above definition of consistency model shifts the focus from the replicated data to the processes using data—so, focussing on the notion of consistency in the *use* of data, based on the application needs

Continuous Consistency

Goal

Imposing limits to deviations between replicas

Continuous Consistency

Goal

Imposing limits to deviations between replicas

Deviations

- numerical deviations—absolute / relative
- staleness deviations—e.g., "fresh" weather reports
- ordering deviations—e.g., distributed updating of replicas, waiting for confirmation

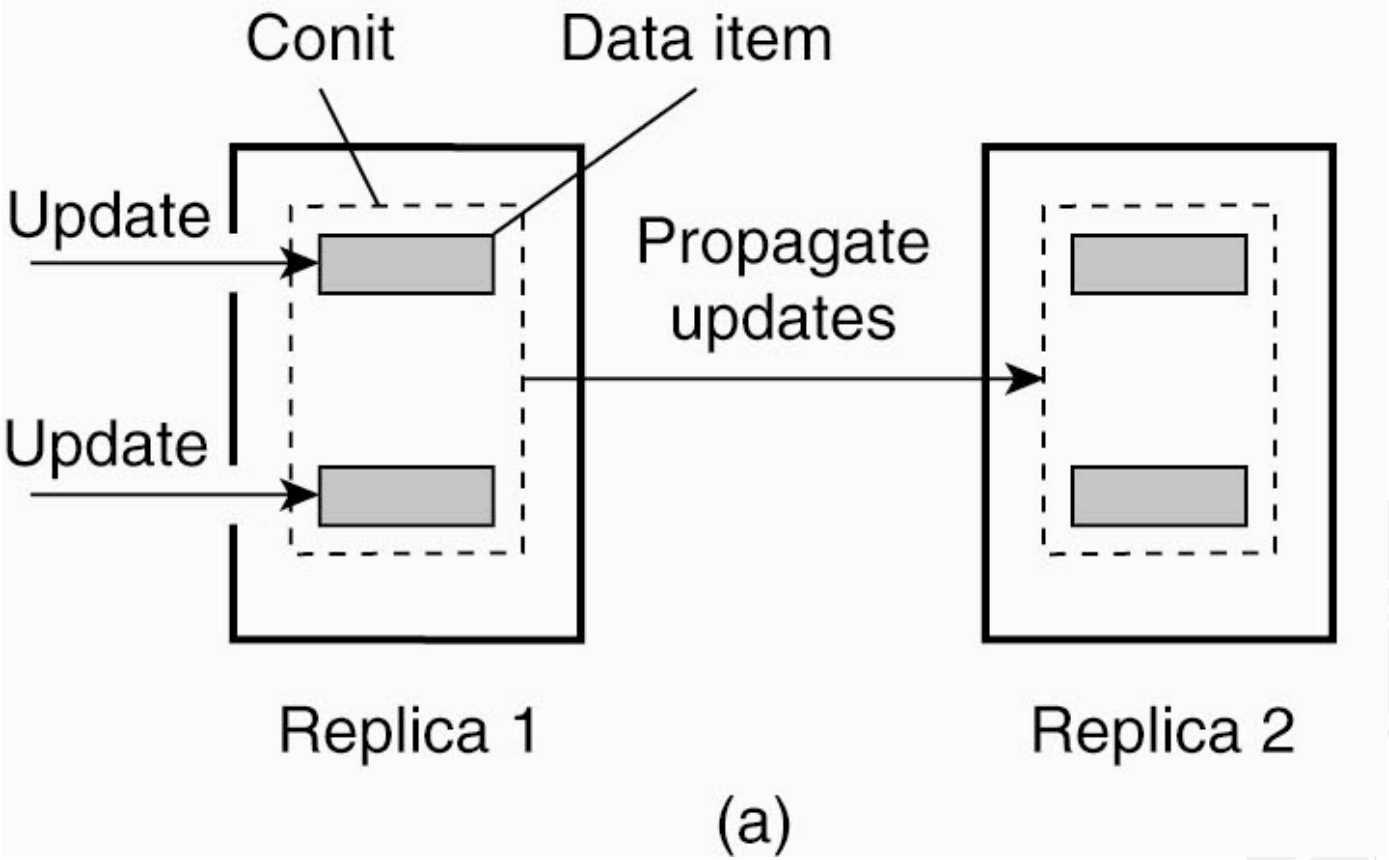
This defines the notion of *continuous consistency*

Inconsistency

Conit

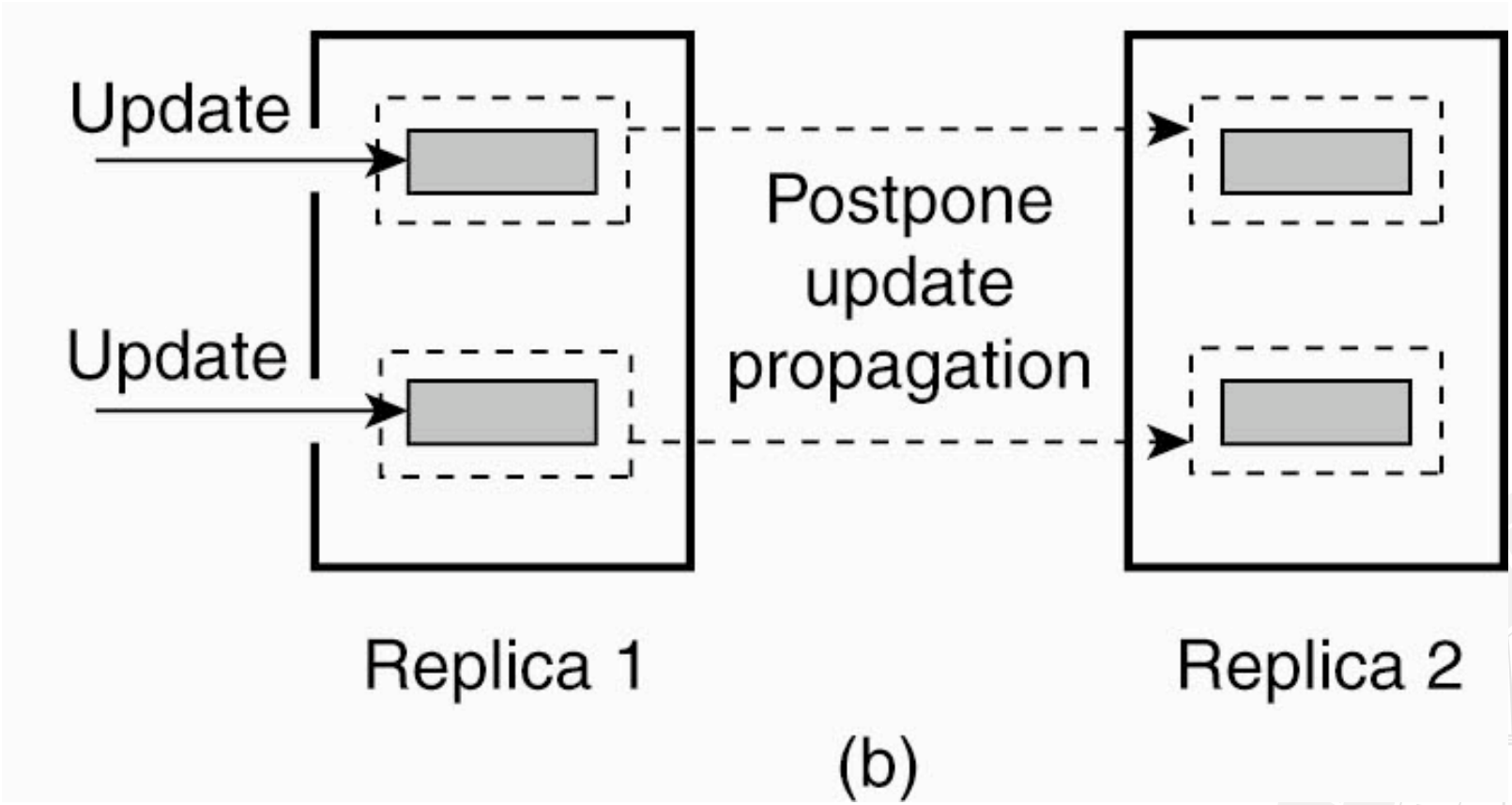
- Notion of *unit of consistency*, called *conit*
- Each data store either implicitly or explicitly suggests its conit
- However, a consistency model (and replication) is defined around a suitably-designed notion of conit
- Deviation is measured in terms of differences of conits

The Choice of Granularity of Conit I



[Tanenbaum and van Steen, 2007] Larger conit

The Choice of Granularity of Conit II



Smaller conit, minor need for propagation
[Tanenbaum and van Steen, 2007]

Consistent Operations Ordering

Main issue

- From parallel and concurrent environments, where several processes share resources, and have to access them simultaneously
- New models conceptually extending data-centric ones: when committing on a state for replicas, an agreement has to be reached among processes upon the global *ordering of updates*

Sequential Consistency

Main idea

- All update operations are seen by all processes in the same order

Sequential Consistency

Main idea

- All update operations are seen by all processes in the same order

Definition

- A data store is *sequentially consistent* when the result of any execution is the same as if the (read and write) operations by all processes on the data store were executed in some sequential order, and the operations of each individual process appear in this sequence in the order specified by its program

Causal Consistency

Main idea

- Weakening sequential consistency
- Based on the notion of cause/effect relation
- Unrelated operations are *concurrent* ones
- Ordering is limited to operations in cause/effect relation

Causal Consistency

Main idea

- Weakening sequential consistency
- Based on the notion of cause/effect relation
- Unrelated operations are *concurrent* ones
- Ordering is limited to operations in cause/effect relation

Definition

- A data store is *causally consistent* when all processes see write operations in cause/effect relation in the same order

Outline

Switching Perspective

Sharing data in mobile computing scenario

- A client connects with different replicas over time
- Differences between replicas should be made transparent
- No particular problems of simultaneous updates, here

Switching Perspective

Sharing data in mobile computing scenario

- A client connects with different replicas over time
- Differences between replicas should be made transparent
- No particular problems of simultaneous updates, here

Client-centric consistency models

- In essence, they ensure that whenever a client connects to a new replica, that replica is up to date according to the previous accesses of the client to the same data in the other replicas on different sites

Eventual Consistency

Scenario

- Large, distributed data store with almost no update conflicts
- Typically, a single authority updating, with many processes simply reading
- The only conflict is *read-write conflict* where one process wants to update a data item while another concurrently attempts to read the same data
- Examples: DNS changes, Web content

Eventual Consistency

Scenario

- Large, distributed data store with almost no update conflicts
- Typically, a single authority updating, with many processes simply reading
- The only conflict is *read-write conflict* where one process wants to update a data item while another concurrently attempts to read the same data
- Examples: DNS changes, Web content

Issues

- Non-updated data may be provided to readers
- In most cases, such an inconsistency might be acceptable to readers
- Typically, if no update takes place for a while, gradually all replicas will become consistent
- This sort of consistency is called *eventual consistency*

Monotonic Reads

Definition

A data store is said to provide *monotonic-read consistency* if the following condition holds

- If a process reads the value of a data item x, any successive read operation on x by the process will always return that same value or a more recent value

Monotonic Reads

Definition

A data store is said to provide *monotonic-read consistency* if the following condition holds

- If a process reads the value of a data item x, any successive read operation on x by the process will always return that same value or a more recent value

Example

- Distributed e-mail database

Monotonic Writes

Definition

A data store is said to provide *monotonic-write consistency* if the following condition holds

- A write operation by a process on a data item x is completed before any successive operation on x by the same process

Monotonic Writes

Definition

A data store is said to provide *monotonic-write consistency* if the following condition holds

- A write operation by a process on a data item x is completed before any successive operation on x by the same process

Idea

- The order of updates is maintained over distributed replicas

Monotonic Writes

Definition

A data store is said to provide *monotonic-write consistency* if the following condition holds

- A write operation by a process on a data item x is completed before any successive operation on x by the same process

Idea

- The order of updates is maintained over distributed replicas

Example

- Software library under development

Read Your Writes

Definition

A data store is said to provide *read-your-writes consistency* if the following condition holds

- The effect of a write operation by a process on data item x will always be seen by a successive read operation on x by the same process

Read Your Writes

Definition

A data store is said to provide *read-your-writes consistency* if the following condition holds

- The effect of a write operation by a process on data item x will always be seen by a successive read operation on x by the same process

Idea

- Avoid the "web page failed update" effect

Read Your Writes

Definition

A data store is said to provide *read-your-writes consistency* if the following condition holds

- The effect of a write operation by a process on data item x will always be seen by a successive read operation on x by the same process

Idea

- Avoid the "web page failed update" effect

Example

- Password updating

Writes Follow Reads

Definition

A data store is said to provide *writes-follow-reads consistency* if the following condition holds

- A write operation by a process on data item x following a previous read operation on x by the same process is guaranteed to take place on the same or a more recent value of x that was read

Writes Follow Reads

Definition

A data store is said to provide *writes-follow-reads consistency* if the following condition holds

- A write operation by a process on data item x following a previous read operation on x by the same process is guaranteed to take place on the same or a more recent value of x that was read

Idea

- Writes affect only up-to-date data items

Writes Follow Reads

Definition

A data store is said to provide *writes-follow-reads consistency* if the following condition holds

- A write operation by a process on data item x following a previous read operation on x by the same process is guaranteed to take place on the same or a more recent value of x that was read

Idea

- Writes affect only up-to-date data items

Example

- Comments to posts on FaceBook

Outline

Key Issue of Replication

Supporting replication in a distributed system

- means deciding where, when and by whom replicas should be placed
- and which mechanisms should be adopted to keep replicas consistent

Key Issue of Replication

Supporting replication in a distributed system

- means deciding where, when and by whom replicas should be placed
- and which mechanisms should be adopted to keep replicas consistent

Two subproblems

- placing *replica servers*
- placing *content*
- not the same problem indeed

Outline

Replicating Services

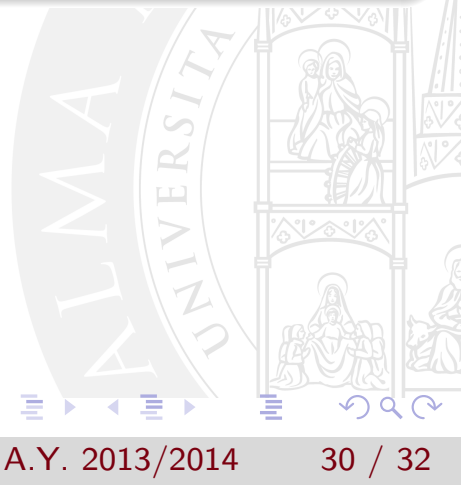
Replication does not mean replicating data—not merely

- services could be replicated as well in a distributed setting
- for the same reasons of data stores
- this essentially means replicating functions, which may / may not insist on the same data store
- two layers for replications, with two consistency & replication models

Replicating Processes

Mobility & cloning for replication

- processes could also be replicated in a distributed mobile setting
- again, for the same reasons of data stores
- this might require cloning...
- ... but also higher-level mechanisms, like goal-passing



References I


Tanenbaum, A. S. and van Steen, M. (2007).
Distributed Systems. Principles and Paradigms.
Pearson Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition.

Consistency & Replication in Distributed Systems

Distributed Systems

Sistemi Distribuiti

Andrea Omicini

andrea.omicini@unibo.it

Dipartimento di Informatica – Scienza e Ingegneria (DISI)

Alma Mater Studiorum – Università di Bologna a Cesena

Academic Year 2013/2014