Consistency

470410-1
Spring 2017
4/20/2017
Kyoung Shin Park
Multimedia Engineering
Dankook University

Consistency & Replication

- Consistency & Replication:
  - What are the reasons for replicating, data, objects, or code?
  - How can we make sure that this replicated data is consistent?
  - What do we do we mean by the term consistency?
    And is there more than one type of consistency?
  - What do we mean by client-centric consistency models?
  - How can we implement the replication of data?
    And what are the advantages and disadvantages of different methods?
  - How can we replicate data and still have it be consistent?
  - Examples of consistent systems with replicated data

Replication

- Replication
  - "A notable latency avoidance technique is ***replication***. Normally, when several processes need access to the same data, the data are transferred back and forth, which is inefficient. With replication, multiple copies of the data are held in different locations, and different processes work with different copies. The idea of replication can be extended to computations; in this case the same computation is run on multiple nodes, which saves the communication of results. A well-known example of replication is ***caching***, the insertion of an additional fast memory level that holds frequently used data. Caching is not only used in the design of basic computer architectures, it is also a technique for speeding up the web. The drawback of any form of replication is the necessity to maintain consistency of the replicated data." *(Leopol01)*

Replication

- Replication
  - "A common requirement when data are replicated is for *replication transparency*. That is, clients should not normally have to be aware that multiple *physical* copies of data exist. As far as clients are concerned, data are organized as individual *logical* objects (or objects) and they identify only one item in each case when they request an operation to be performed. Furthermore, clients expect operations to return only one set of values. This is despite the fact that operations may be performed upon more than one physical copy in concert. *(CoDoKi01)*
  - "The other general requirement for replicated data -- one that can vary in strength between applications -- is that of *consistency*. This concerns whether or not the operations performed upon a collection of replicated objects produce results that meet the specification of correctness for those objects." *(CoDoKi01)*

Replication

- Replication
  - "The fundamental problem in managing replicated data is to maintain the consistency of the data. In a local sense, a query on the data should return the data value that was 'most recently written'. In a global sense, the interaction of a program with the collection of all global data should obey a global consistency constraint. "A primary motivation for replicating data is *fault tolerance*." *(ChoJoh97)*

Advantages of Replication

- The advantages of replication include the following (Sinha97):
  - "Increased availability"
  - "Increased reliability"
  - "Improved response time"
  - "Reduced network traffic"
  - "Improved system throughput"
  - "Better scalability"
  - "Autonomous operation"

Replication versus Caching

- "Replication is often confused with caching, probably because they both deal with multiple copies of a data. However, the two concepts have the following basic differences:
  1. "A replica is associated with a server, whereas a cached copy is normally associated with a client.
  2. "The existence of a cached copy is primarily dependent on the locality in file access patterns, whereas the existence of a replica normally depends on availability and performance requirements.
  3. "As compared to a cached copy, a replica is more persistent, widely known, secure, available, complete, and accurate.
  4. "A cached copy is contingent upon a replica. Only by periodic revalidation with respect to a replica can a cached copy be useful." *(Sinha97)*

Consistency

- Consistency
  - "***Consistency*** is more difficult to achieve in a distributed system. The lack of global information, potential replication and partitioning of data, the possibility of component failures, and the complexity of interaction among modules all contribute to the problem of inconsistency in the system. **A system is consistent from the user's perspective if there is uniformity in using the system and the system behavior is predictable.** In addition, the system must be capable of maintaining its integrity with proper concurrency control mechanisms and failure handling and recovery procedures. Consistency control in data and files (or database in a transaction-oriented system) is a crucial issue in distributed file systems." *(ChoJoh97)*

Consistency

- Consistency
  - "***Consistency*** is a term used to describe the function of ensuring all data copies are the same and correct. There are three popular methods for realizing cache consistency. The first employs software to enforce critical regions, protected regions of code where a given process is changing shared data ... . The second utilizes software to prevent a processor from ever caching shared memory. The third method for maintaining cache consistency is referred to as ***snoopy cache*** ... every processor constantly snoops or monitors the shared bus, relying on the fact that all processors are connected via a common bus." *(Galli00)*

Consistency

- Consistency
  - "A ***consistency model*** basically refers to the degree of consistency that has to be maintained for the shared-memory data for the memory to work correctly for a certain set of applications. It is defined as a set of rules that applications must obey if they want the DSM system to provide the degree of consistency guaranteed by the consistency model." *(Sinha97)*

Type of Consistency

- ***Strict*** Consistency Model:
  - "The *strict consistency model* is the strongest form of memory coherence, having the most stringent consistency requirements. A shared-memory system is said to support the strict consistency model if the value returned by a read operation on a memory address is always the same as the value written by the most recent write operation to that address, irrespective of the locations of the processes performing the read and write operations. That is, all writes instantaneously become visible to all processes." *(Sinha97)*

Type of Consistency

- ***Sequential*** Consistency Model:
  - "The *sequential consistency model* was proposed by Lamport ... . A shared-memory system is said to support the sequential consistency model if all processes see the same order of all memory access operations on the shared memory. The exact order in which the memory access operations are interleaved does not matter. ... If one process sees one of the orderings of ... three operations and another process sees a different one, the memory is not a sequentially consistent memory." *(Sinha97)*

Type of Consistency

- ***Casual*** Consistency Model:
  - "The *causal consistency model* ... relaxes the requirement of the sequential model for better concurrency. Unlike the sequential consistency model, in the causal consistency model, all processes see only those memory reference operations in the same (correct) order that are potentially causally related. Memory reference operations that are not potentially causally related may be seen by different processes in different orders."
    (Sinha97)

Type of Consistency

- ***FIFO*** Consistency Model:
  - For *FIFO consistency*, "Writes done by a single process are seen by all other processes in the order in which they were issued, but writes from different processes may be seen in a different order by different processes.
    "FIFO consistency is called *PRAM consistency* in the case of distributed shared memory systems."
    (Tanvan02)

Type of Consistency

- ***Pipelined Random-Access Memory (PRAM)*** Consistency Model:
  - "The *pipelined random-access memory (PRAM) consistency model* ... provides a weaker consistency semantics than the *(first three)* consistency models described so far. It only ensures that all write operations performed by a single process are seen by all other processes in the order in which they were performed as if all the write operations performed by a single process are in a pipeline. Write operations performed by different processes may be seen by different processes in different orders."
    (Sinha97)

Type of Consistency

- ***Weak*** Consistency Model:
  - "Synchronization accesses (accesses required to perform synchronization operations) are sequentially consistent. Before a synchronization access can be performed, all previous regular data accesses must be completed. Before a regular data access can be performed, all previous synchronization accesses must be completed. This essentially leaves the problem of consistency up to the programmer. The memory will only be consistent immediately after a synchronization operation."
    (SinShi94)

Type of Consistency

- ***Release*** Consistency Model:
  - "*Release consistency* is essentially the same as weak consistency, but synchronization accesses must only be processor consistent with respect to each other. Synchronization operations are broken down into *acquire* and *release* operations. All pending acquires (e.g., a lock operation) must be done before a release (e.g., an unlock operation) is done. Local dependencies within the same processor must still be respected.
    "Release consistency is a further relaxation of weak consistency without a significant loss of coherence."
    (SinShi94)

Type of Consistency

- ***Entry*** Consistency Model:
  - "Like ... variants of release consistency, it requires the programmer (or compiler) to use acquire and release at the start and end of each critical section, respectively. However, unlike release consistency, *entry consistency* requires each ordinary shared data item to be associated with some synchronization variable, such as a lock or barrier. If it is desired that elements of an array be accessed independently in parallel, then different array elements must be associated with different locks. When an acquire is done on a synchronization variable, only those data guarded by that synchronization variable are made consistent."
    (Tanvan02)

Type of Consistency

- ***Processor*** Consistency Model:
  - "Writes issued by a processor are observed in the same order in which they were issued. However, the order in which writes from two processors occur, as observed by themselves or a third processor, need not be identical. That is, two simultaneous reads of the same location from different processors may yield different results."
    (SinShi94)
- ***General*** Consistency Model:
  - "A system supports *general consistency* if all the copies of a memory location eventually contain the same data when all the writes issued by every processor have completed."
    (SinShi94)

ACID (Atomicity, Consistency, Isolation, Durability)

- Related to the consistency of a distributed system (or database system) are the ***ACID*** properties
- The ***ACID (Atomicity, Consistency, Isolation, Durability) properties*** are primarily concerned with achieving the ***concurrency transparency*** goal of a distributed system
  - a property that allows sharing of objects without interference.
  - In a sense, the execution of a transaction appears to take place in a critical section. However, operations from different transactions are interleaved (in some 'safe' way) to gain more concurrency.

ACID (Atomicity, Consistency, Isolation, Durability)

- ***Atomicity***.
  - Either all of the operations in a transaction are performed or none of them are, in spite of failures.
- ***Consistency***.
  - The execution of interleaved transactions is equivalent to a serial execution of the transactions in some order.
- ***Isolation***.
  - Partial results of an incomplete transaction are not visible to others before the transaction is successfully committed.
- ***Durability***.
  - The system guarantees that the results of a committed transaction will be made permanent even if a failure occurs after the commitment.

References

- http://www.cs.colostate.edu/~cs551/CourseNotes/Consistency/ReplIndex.html