CompSci131

Parallel and Distributed Systems

Prof. A. Veidenbaum

Today's topics

- Consistency and replication
- Consistency models

- Reading assignment:
  - Today: 7.2
  - Next time: The rest of 7.2

Last Lecture Covered

- **Uses of epidemic protocols**
- **The consistency problem**
- **Continuous consistency**

Consistency

- **Existence of multiple replicas in a data store may lead to replicas having different "values" - *an inconsistency***
  - **Any use of an inconsistent value may lead to program errors**
  - ***Strict consistency is impossible to achieve in DS***

- **A consistency unit (conit) is a data unit on which consistency is defined**
  - **An "update unit": a byte, a word, a cache line, a page, etc**

- **Use of conit can lead to "false sharing"**
  - **Conit contains multiple independently used data items**
    - » **Means consistency really did not have to be enforced at this level of granularity**

Notation

- **Will define a Read set and a Write set at process i**

- **$W_i(x)a$: <u>process *i*</u> writes the value <u>*a*</u> to variable <u>*x*</u>**
  - **Data_Store[x] = *a***

- **$R_i(x)b$: process *i* reads *x*, Data_Store[x] returns a value *b***
  - **Initial value for data at an "address" is NIL**

- **This allows us to define Write/Read ordering and will help figuring out consistency**

- **Have been talking about read/write ordering for a single item – Read(x) / Write(x)**
  - **Consistency is more than that, it also covers ordering of Rd/Wr for multiple items together**

(Cache) Coherence

- A consistency model defines ordering of accesses to all *data items* in the data store
  - *conit's*

- Coherence defines what happens on access to a single data item
  - A conit = cache line (cache block)
  - Can also have "false sharing"

- An example of false sharing

- In summary, want to achieve a consistent ordering of operations on shared, replicated data by all participating processes

Relaxing Consistency Requirements

- **Programmers can reason about concurrent programs even if the system is not strict**

- **For instance, the following is often assumed:**
  - **Concurrent programs should not assume anything about relative speed of processes**
    - **Thus cannot know about event order**
  - **Synchronization <u>must</u> be used to achieve a certain event order**

- ***What other assumptions can be made that would still allow programmers to reason about concurrent or distributed programs?***

Sequential Consistency

- **Sequential Consistency was defined by Lamport for multiprocessors**

- **Definition**
  1. **The result of any execution is the same as if all Rd/Wr operations by all processes were executed in some sequential order**

     AND

  2. **The operations of each individual process appear in this sequence in the order specified by its program**

- **In other words, the result of an execution may be any legal interleaving of accesses**
  - **Does not say that a read gets the result of most recent write**

- ***But all processes MUST see the same order***

- **No reference to time!**

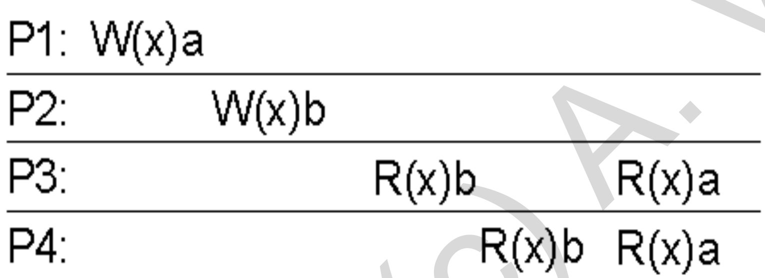
Sequential Consistency Examples

a) Is it a sequentially consistent data store?

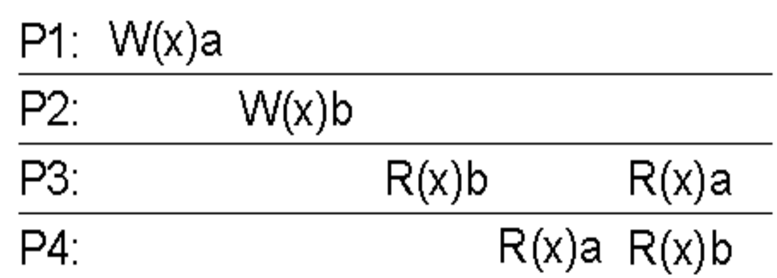
b) What about this one?

Why?

- Time is the horizontal axis
  - Is it important?



(a)



(b)

Let’s look a little deeper

- **3 data items: x,y,z. Initialized to 0.**
- **Three concurrently executing processes**
  - **6! Possible inter-leavings of the operations**
  - **Some are valid and some violate program order!**
    - » **Print before assignment**
- **A signature: values printed by processes P1,P2,P3**
    - » **In this order**
      - **00 11 01**

Process P1	Process P2	Process P3
x = 1;	**y = 1;**	**z = 1;**
print (y, z);	**print (x, z);**	**print (x, y);**

4 Valid Execution Sequences

- There are 2^6 possible *signatures* in this case
  - Not all of these are valid!
    - » 000000?
- Overall, many different ordering are perfectly legal!

(a)	(b)	(c)	(d)
x = 1;	x = 1;	y = 1;	y = 1;
print (y, z);	y = 1;	z = 1;	x = 1;
y = 1;	print (x,z);	print (x, y);	z = 1;
print (x, z);	print(y, z);	print (x, z);	print (x, z);
z = 1;	z = 1;	x = 1;	print (y, z);
print (x, y);	print (x, y);	print (y, z);	print (x, y);
Prints: 001011	Prints: 101011	Prints: 010111	Prints: 111111
Signature: 001011	Signature: 101011	Signature: 110101	Signature: 111111

Reasoning About Consistency

- The example showed that multiple orderings are sequentially consistent
  - But not all
- A program should work with any valid order
  - A program that works for some but not all is incorrect
    - » It violates the contract with the data store
- Program order deals with how accesses to different data items are interleaved
- Data coherence looks at access to the same item