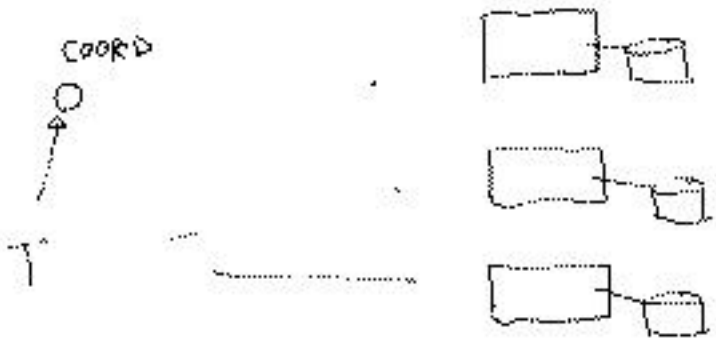
Special Case:

Replicated Databases



All are the same
(so all updates are
the same)

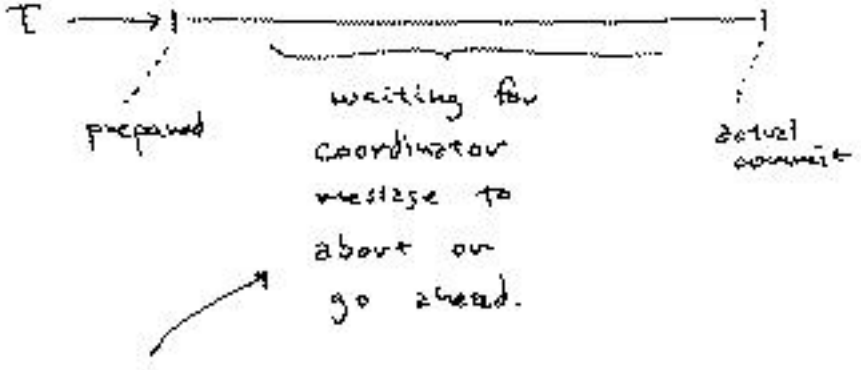
How does this

Simplify the problem?

- All updates & commits are the same every where!
- Each commit can be done by "barrier synchronization"

Two Phase Commit only works if each transaction manager (host) can guarantee that a "prepared" transaction will be committed.

How to do this?

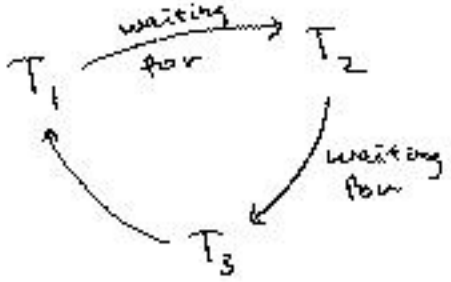


during this time, transaction manager blocks all other transactions with conflicting read/write sets

— done by locks on items

Hence, distributed deadlock is possible.

In single-host systems, deadlock is detected by finding a cycle in a "wait-for" graph



The transaction (lock) manager can build this graph as it processes lock/unlock requests.

Ex: T requests write lock (X), but T', T'' already have read locks on X.

this is added to the wait for graph.

But this doesn't find distributed deadlocks!

T_2, T_1 is running on both A & B

How can we detect such deadlocks?

(Class Suggestions)