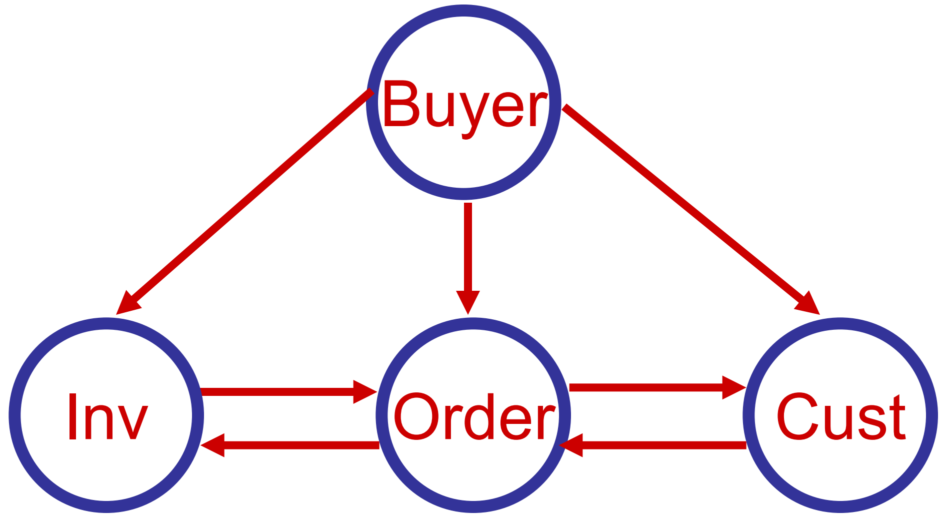
2 Phase Commit -> 3PC

Intel (TX memory):
Transactional
Synchronization
Extensions (TSX)

Trans in Distributed Systems

- A distributed transaction involves
  - updates at multiple nodes
  - and the messages between those nodes
- For example, buying widgets



Distributed Atomic Commit Requirements

1. All workers that reach a decision reach the same one
2. Workers cannot change their decisions on commit or abort once a decision is made
3. To commit all workers must vote commit
4. If all workers vote commit and there are no failures the transaction will commit
5. If all failures are repaired and there are no more failures each worker will eventually reach a decision (In fact it will be the same decision)

Two phase commit variants

- Centralized 2PC: workers only communicate with the coordinator
- Linear 2PC: coordinator, and all workers in a single line/chain
- Decentralized 2PC: all workers can communicate with one another

Process uncertainty in atomic commit

- Uncertainty period for a process
  - Time between the moment a process votes Yes (commit) and the moment it knows the txn decision (tx-abort or tx-commit)

- While process is uncertain it is blocked: process cannot make progress
- Blocking also arises when process must wait for failures to be repaired before proceeding

Hard failure constraints on distributed atomic commit with failures

- A non-blocking distributed atomic commit protocol that handles node failures and communication failures is impossible (i.e., none can exist)

- Cannot solve it with communication failures. Why?

Hard failure constraints on distributed atomic commit with failures

- In general, a non-blocking distributed atomic commit protocol that handles node failures and communication failures is impossible (i.e., none can exist)
- Cannot solve it with communication failures. Why?
  - Cannot eliminate uncertainty periods with comm. failures: process has to cast vote AND learn all other votes simultaneously!
- Therefore, any ACP (atomic commit protocol) may cause processes to become blocked during communication failures

Hard failure constraints on distributed atomic commit with failures

- In general, a non-blocking distributed atomic commit protocol that handles node failures and communication failures is impossible (i.e., none can exist)

- 2PC: can block in both cases (examples?)
  - And we saw that 2PC topology does not matter
- 3PC: solves atomic distributed commit with node failures

  (but not communication failures)

2PC is a blocking protocol

- Coordinator could fail after having decided the outcome, which would lead all worker nodes to block
  - Key issue: If all nodes are uncertain, then they are blocked

2PC is a blocking protocol

- Coordinator could fail after having decided the outcome, which would lead all worker nodes to block
  - Key issue: If all nodes are uncertain, then they are blocked

- 3PC: solves atomic distributed commit with node failures (but not communication failures)
- How? 3PC satisfies the following key condition:
- Cond: if any operational node is uncertain then no process (operational or failed) can have decided to Commit.
  - i.e., if working node discovers it is uncertain, it can decide to abort: no blocking!

Why 2PC not satisfy cond

- Coord sends tx-commit to p,q
  - p receives tx-commit before q
  - p will decide to commit before q (which is uncertain)
  - i.e., it’s a kind of a race condition!

How 3PC solves this

- Coord sends pre-commit messages if all votes were to commit
- When worker receive a pre-commit it knows that all participants voted to commit. But, it does not commit at this time
- Each worker acks the pre-commit
- Coord receipts acks, and when all recvd, knows no node is uncertain
- At this point it decides commit and sends a tx-commit

How 3PC solves this

- Note: acks from nodes and tx-commit from coord is known to nodes ahead of time! Weird..?
- Their purpose is to signal events, not to communicate info
  - Receipt of ack from p: tells coord that p is not uncertain
  - Receipt of tx-commit at p: tells p that that no worker is uncertain
  - This last statement is key: it allows p to commit without violating Cond

Termination rules

- Okay, but how does 3PC handle a coord failure?
- New coordinator boots up and must complete any outstanding transactions, using collected process states:
  - TR1: if someone is aborted, decide abort
  - TR2: if some is committed, decide commit
  - TR3: if all uncertain, decide abort
  - TR4: If some committable, but none committed, do another round of pre-commit, get acks, then decide commit.