



Atomic Commit

Coordonarea tranzacțiilor distribuite

Problema commit-ului în sisteme distribuite

- O singură tranzacție poate implica mai multe noduri
- Fiecare nod are o stare proprie (bază de date, memorie)
- Comunicarea se face prin rețea nesigură:
 - Mesaje întârziate sau pierdute
- Nodurile pot cădea sau reporni independent
- Fără un mecanism special:
 - Unele noduri pot face commit
 - Altele pot face abort
- Rezultatul este inconsistența sistemului



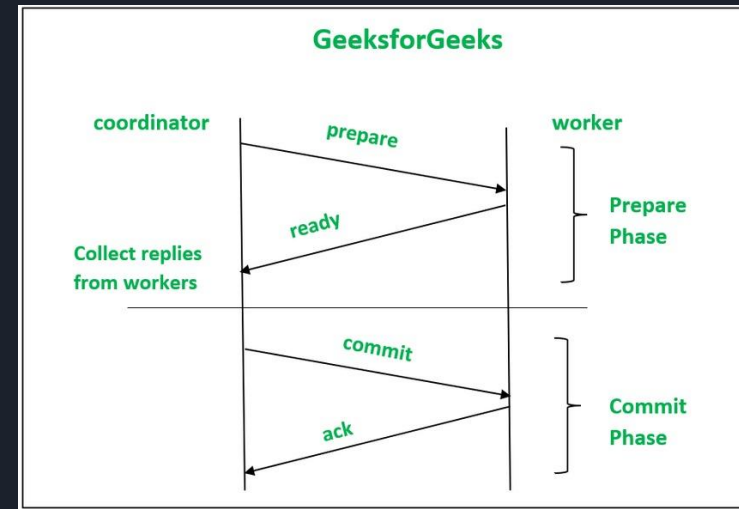
Atomic Commit: definiție și proprietăți

- Atomic commit este o problemă de coordonare în sistemele distribuite
- Se aplică tranzacțiilor distribuite
- O tranzacție are o singură decizie globală
- Rezultatul posibil este:
 - Commit pe toate nodurile
 - Abort pe toate nodurile
- Proprietăți:
 - Atomicitate -> toate modificările sau niciuna
 - Acord -> toate nodurile decid la fel
 - Validitate -> dacă toate votează YES, decizia este commit
 - Terminație -> protocolul ajunge la o decizie

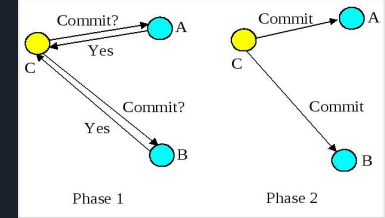


Modelul clasic: Two-Phase Commit (2PC)

- Protocol clasic pentru rezolvarea problemei de Atomic Commit
- Asigură decizie globală: toate nodurile commit sau toate abort
- Implică 2 tipuri de noduri:
 - Coordinator -> coordonează tranzacția și ia decizia finală
 - Participants -> noduri care execută modificările
- Coordinator:
 - Inițiază protocolul
 - Colectează voturile participanților
 - Decide commit sau abort
- Participants (workers):
 - Verifică dacă pot aplica modificările
 - Votează YES/NO
 - Respectă decizia coordinator-ului



Two-Phase Commit: pașii protocolului



- Faza 1: Prepare (Voting Phase)

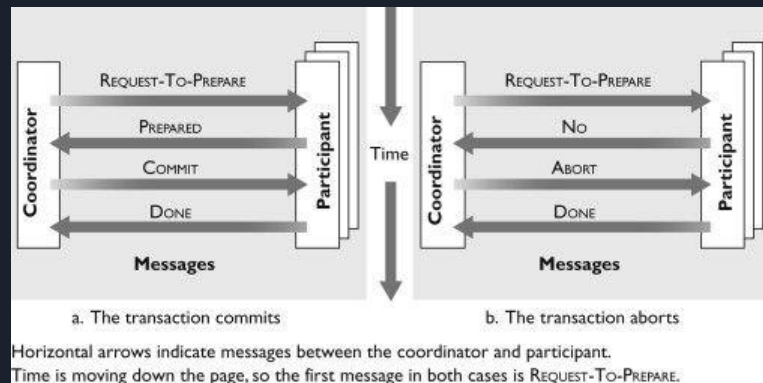
- Coordonatorul trimite mesaje PREPARE tuturor participanților
- Fiecare participant:
 - Verifică dacă poate aplica modificările
 - Blochează resursele necesare
 - Scrie starea în log (pentru recovery)
- Participantul răspunde
 - Yes/Ready -> poate face commit
 - No - nu poate continua

- Faza 2: Commit/Abort

- Dacă toți participanții au răspuns YES:
 - Coordonatorul trimite COMMIT
- Dacă cel puțin unul răspunde NO:
 - Coordonatorul trimite ABORT
- Participanții:
 - Aplică decizia finală
 - Eliberează resursele
 - Trimit ACK către coordonator

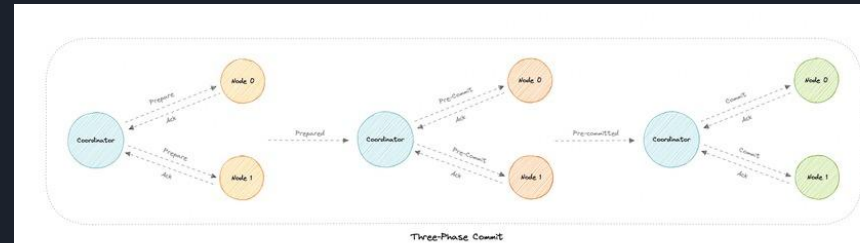
Probleme și limitări ale Two-Phase Commit

- Protocolul nu este fault-tolerant complet
- Principala problemă: blocking
- Situație critică:
 - Participanții au votat YES
 - Coordonatorul cade înainte de decizia finală
- Participanții:
 - Nu pot decide singuri
 - Trebuie să aștepte coordonatorul
 - Rămân blocați pe termen nedefinit
- Timeout-ul nu ajută:
 - Nu pot face abort în siguranță
- Impact:
 - Resurse blocate
 - Performanță scăzută
 - Disponibilitate redusă



Three-Phase Commit (3PC): reducerea blocking-ului

- Variantă propusă pentru a reduce limitările Atomic Commit în 2PC
- Scop principal: evitarea blocking-ului
- Faze:
 - CanCommit? -> participanții pot sau nu să continue
 - PreCommit -> participanții sunt informați că urmează commit-ul
 - DoCommit/Abort -> decizia finală
- Participanții pot lua decizii locale bazate pe timeout-uri
- Funcționează doar sub anumite ipoteze:
 - Sistem aproape sincron
 - Timeouts fiabile
 - Crash-uri detectabile





Atomic Commit prin consens distribuit

- Decizia de commit/abort poate fi luată prin consens
- Algoritmi de consens: Paxos, Raft
- Model:
 - Un leader
 - Un log replicat
 - Decizia de commit este o intrare în log
- Toate nodurile aplică decizia după ce logul este replicat
- Avantaje față de 2PC:
 - Toleranță mai bună la crash-uri
 - Evită blocarea permanentă
- Dezavantaj:
 - Complexitate și cost mai mari
-



Implementări ale Atomic Commit și trade-offs

- Two-Phase Commit (2PC):
 - Implementare clasică a Atomic Commit
 - Simplă, poate bloca
- Three-Phase Commit (3PC):
 - Extensie a Atomic Commit
 - Reduce blocarea, ipoteze stricte
- Atomic commit prin consens:
 - Decizia este replicată prin acord global
 - Robust la crash-uri
 - Cost și complexitate mai mari
- Context de utilizare:
 - Baze de date distribuite
 - Sisteme tightly-coupled
- Criterii de alegere:
 - Latență
 - Disponibilitate
 - Toleranță la erori
 - Complexitate



Bibliografie

- [Wikipedia, Atomic Commit, 15 September 2023, at 10:27 \(UTC\)](#)
- [Karan Pratap Singh, Distributed Transactions, 2025](#)
- [Nathan VanBenschoten, Parallel Commits: An atomic commit protocol for globally distributed transactions, November 7, 2019](#)
- [Annie Ahuja, Atomic Commit Protocol in Distributed System, 23 Jul, 2025](#)
- [Charlie Custer, Distributed transactions: What, why, and how to build a distributed transactional application, February 16, 2023](#)
- [Krishna Ranjan, Commit Protocol in DBMS, 07 Mar, 2024](#)