Announcements

- Hard deadline for survey paper and project report: Wednesday, December 8, 8:59am by email
- Final presentations: December 13-15
- For projects only:
  - submit code (tarball + README suffices) before final presentation
  - Either:
    - Include README telling me how to run your project
    - Or schedule demo with me individually

Outline for Today

- Finish discussion of distributed transactions
  - Local transactions
  - Distributed protocols

Recap

- Transactions (ACID)
  - Atomicity
  - Consistency
  - Isolation
    - (Serializable)
  - Durability

Implementing Transactions

- Achieving Atomicity
  - Private workspace
  - Writeahead Log
- Achieving Consistency + Isolation
  - Concurrency Control (Locking or Timestamps)
- Achieving Durability
  - Implemented by writing to stable or persistent storage

Private Workspace
Writeahead Log

\begin{align*}
x &= 0; & \text{Log} & \quad x = 0 / 1 \\
y &= 0; & \text{Log} & \quad y = 0 / 2 \\
\text{BEGIN\_TRANSACTION;} \quad x &= x + 1; & \text{Log} & \quad [x = 0 / 1] \\
\text{y} &= y + 2; & \text{Log} & \quad [y = 0 / 2] \\
\text{x} &= y \times y; & \text{Log} & \quad [x = 1 / 4] \\
\text{END\_TRANSACTION;} \quad (a) & \quad (b) & \quad (c) & \quad (d)
\end{align*}

After each operation, add [variable = old/new] to log.

Scheduler & Concurrency Control

Implementing Serializability

\begin{enumerate}
\item Consider read/write operations
  \begin{itemize}
  \item read/write conflict
  \item write/write conflict
  \end{itemize}
\item Goal: create serialization order (explicit or implicit)
  \begin{itemize}
  \item Locking or Timestamp-based scheme
  \end{itemize}
\item Optimistic vs. Pessimistic
\end{enumerate}

Pessimistic 2-Phase-Locking

Strict 2-Phase-Locking

Timestamp-based Ordering

\begin{itemize}
\item Pessimistic version
\item Alternative: associate totally-ordered timestamps with transactions and reads/writes (a la Lamport)
\item Stamp reads+writes with transaction’s TS
\item Abort read if encountering larger ("later") write timestamp ("too late")
\item Abort write if encountering later read or write timestamp
\item Does not suffer from deadlock! (unlike 2PL)
\end{itemize}
Distributed Transactions

- Consistency + Isolation
  - Distributed locking
  - 2PL is problem of distributed mutual exclusion
- Atomicity + Durability
  - Can achieve local atomicity & durability
  - Need distributed commit

2-Phase-Commit (Gray 1978)

INIT
Coordinator

Process 1
INIT

Process 2
INIT

Process 3
INIT

COMMIT
Coordinator

Process 1
COMMIT

Process 2
COMMIT

Process 3
COMMIT
2-Phase-Commit (Abort)

Process 1
INIT

Process 2
INIT

Process 3
INIT

2-Phase-Commit (Abort)

Coordinator

Process 1
WAIT

Process 2
Vote Request

Process 3
Vote Request

2-Phase-Commit (Abort)

Coordinator

Process 1
ABIORT

Process 2
GLOBAL Abort

Process 3
GLOBAL Abort

2PC Discussion

- How is atomicity provided?
- What are effects of coordinator failure?
  - How can participants help each other?
  - If at least one is in COMMIT; is in INIT?
  - What if all are “READY”?

3 Phase Commit

(a) Central Coordinator
(b) Participant

State Machine for 2PC

(a) Central Coordinator
(b) Participant

State Machine for 2PC

(a) Central Coordinator
(b) Participant
Summary

• Transactions
  – ACID
  – 2PhaseLocking
• Distributed Transactions
  – 2PhaseCommit

Thursday:
  – Multitasking in the Java virtual machine (MVM)
  – Paravirtualization (Xen)