Distributed Systems

Kai Shen

Parallel Computing vs. Distributed Systems

- Parallel computing – compute sub-tasks simultaneously so that work can be completed faster.
- Distributed systems – a set of autonomous computers working together to appear as a single coherent system (to achieve unified goals).
  - World wide web, networked file system, instant messaging, …
- Parallel computing more identified with what it does
- Distributed systems more identified with what they are
- Not mutually exclusive

Distributed Systems Model

- Study of general approaches requires a system model
- Autonomous computers
- Network connecting all computers (all-to-all reachable)
- Communications
  - Messages
  - Streams
- Failures
  - Messages may be lost
  - Nodes may stop working
  - Or worse!

Distributed Systems Overview

- Time and order
- Fault tolerance
- Replication and consistency
- Scalable Internet systems (data centers, cloud computing)

Time and Clocks

- Why is it important?
  - Determine the order of events occurred on different computers
- Examples:
  - In "make", source files who have not been changed since last compile don't have to be recompiled
    - Compilation and editing on different computers
  - Two distributed updates to the same location need to be ordered to know the final state
  - More examples?

Physical Clocks

- How do we get time?
- Fixed-frequency events
  - Quartz crystal oscillates at fixed frequency to triggered timer interrupts
  - Atom vibrates (makes state transitions) at fixed frequency
- Consistent for a single timing device
- But drifts/skews common over multiple devices in a distributed system

Physical Clock Synchronization

- Clock synchronization
  - Identify clock skewness and correct it by adjusting clock
  - No turning back in time
- Cristian's algorithm
  - A client requests time from a server, the server responds with current time
  - Delay in response - estimate as half of request/response time
- What if we know (or can model) the precise delay?
  - GPS

Physical Clock Synchronization

- Averaging of everyone's time
- A central server polls everybody and does the averaging
- Distributed averaging
  - Everyone broadcasts its time periodically
  - Everyone also monitors these broadcasts, calculates clock drifts and averages them

Logical Ordering

- Agreement on ordering of events (rather than the absolute time) is what matters
- Lamport ordering of distributed events
  - a → b, or "a happens-before b" holds regardless of processor speed and message delays
  - Specifically, a happens-before b
    - if a occurs before b on the same computer or a/b are send/receive events of the same message on two computers
    - transitive relation
- Partial ordering

Lamport Logical Clock

- Assigning timestamp to each event C(e) that doesn't violate the "happens-before" partial ordering
- Every computer maintains a local incrementing timestamp
- Each message carries the sending time of the sender
- Upon receipt of a message, the receiver clock is set to the greater of its own or the message timestamp + 1

Utilization in Totally Ordered Broadcasts

- Totally ordered broadcasts
  - Update management in replicated databases
- Assumptions
  - FIFO messages between same sender/receiver pair
  - Messages are not lost
- Solution
  - Each message carries sender's timestamp
  - Received broadcast messages are buffered, acknowledged (in broadcast)
  - Deliver a message if
    - it has the earliest timestamp in buffer
    - and already acknowledged by everyone
  - Guarantee: will not receive a new message with earlier timestamp

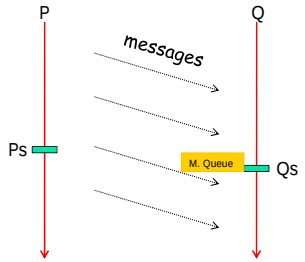
Vector Timestamps

- "happens-before" is a partial ordering
- Lamport logical clock is a total ordering that may go beyond the "happens-before" ordering
- Vector timestamp: n-element vector for n-computer system
- At computer i:
  - VT[i] indicates the number of local events occurred so far (incremented after each local event)
  - VT[j≠i] = k means that computer i knows that k events have occurred at computer j (updated after each received message)
- Each message carries the full timestamp, updates receiver

Utilization on Race Detection

- A distributed race condition may occur if two conflicting updates are not causally ordered

Distributed Snapshots



- Can there be a message sent after Ps and received before Qs?
- What should be in the message queue for Qs?
- Chandy-Lamport algorithm