CLOUD COMPUTING CONCEPTS

with Indranil Gupta (Indy)

PAXOS

Lecture A

THE CONSENSUS PROBLEM

Give it a thought

Have you ever wondered why distributed server vendors always only offer solutions that promise five-9's reliability, seven-9's reliability, but never 100% reliable?

The fault does not lie with the companies themselves, or the worthlessness of humanity.

The fault lies in the impossibility of consensus.

What is common to all of these?

A group of servers attempting:

- To make sure that all of them receive the same updates in the same order as each other
- To keep their own local lists where they know about each other, and when anyone leaves or fails, everyone is updated simultaneously
- To elect a leader among them, and let everyone in the group know about it
- To ensure mutually exclusive (one process at a time only) access to a critical resource like a file

What is common to all of these?

A group of servers attempting:

- Make sure that all of them receive the same updates in the same order as each other [Reliable Multicast]
- To keep their own local lists where they know about each other, and when anyone leaves or fails, everyone is updated simultaneously [Membership/Failure Detection]
- Elect a leader among them, and let everyone in the group know about it [Leader Election]
- To ensure mutually exclusive (one process at a time only) access to a critical resource like a file [Mutual Exclusion]

So what is common?

- Let's call each server a "process" (think of the daemon at each server)
- All of these were groups of processes attempting to *coordinate* with each other and reach *agreement* on the value of something
  - The ordering of messages
  - The up/down status of a suspected failed process
  - Who the leader is
  - Who has access to the critical resource
- All of these are related to the *Consensus* problem

What is Consensus?

Formal problem statement

•N processes

•Each process p has

input variable xp : initially either 0 or 1

output variable yp : initially b (can be changed only once)

•Consensus problem: design a protocol so that at the end, either:

1. All processes set their output variables to 0 (all-0's)
2. Or all processes set their output variables to 1 (all-1's)

What is Consensus? (2)

- Every process contributes a value
- *Goal is to have all processes decide same (some) value*
  - Decision once made can't be changed
- There might be other constraints
  - Validity = if everyone proposes same value, then that's what's decided
  - Integrity = decided value must have been proposed by some process
  - Non-triviality = there is at least one initial system state that leads to each of the all-0's or all-1's outcomes

Why is it Important?

- Many problems in distributed systems are equivalent to *(or harder than)* consensus!
  - Perfect Failure Detection
  - Leader election (select exactly one leader, and every alive process knows about it)
  - Agreement (harder than consensus)

- So consensus is a very important problem, and solving it would be really useful!

- So, is there a solution to Consensus?

Two Different Models of Distributed Systems

- Synchronous System Model and Asynchronous System Model
- Synchronous Distributed System
  - Each message is received within bounded time
  - Drift of each process' local clock has a known bound
  - Each step in a process takes lb < time < ub

  E.g., A collection of processors connected by a communication bus, e.g., a Cray supercomputer or a multicore machine

Asynchronous System Model

- Asynchronous Distributed System
  - No bounds on process execution
  - The drift rate of a clock is arbitrary
  - No bounds on message transmission delays

  E.g., The Internet is an asynchronous distributed system, so are ad-hoc and sensor networks

- *This is a more general (and thus challenging) model than the synchronous system model. A protocol for an asynchronous system will also work for a synchronous system (but not vice-versa)*

Possible or Not

- In the synchronous system model
  - Consensus is solvable

- In the asynchronous system model
  - Consensus is impossible to solve
  - Whatever protocol/algorithm you suggest, there is always a worst-case possible execution (with failures and message delays) that prevents the system from reaching consensus
  - Powerful result (see the FLP proof in the optional lecture of this series)
  - Subsequently, safe or probabilistic solutions have become quite popular to consensus or related problems.

So what next?

- Next lecture: Let’s just solve consensus!