CMPSCI 677 Operating Systems **Spring 2018**

Lecture 12: March 09

Lecturer: Prashant Shenoy *Scribe:* ***Abhiram Eswaran, Ping Lin(2017)***

12.1 Overview

The topic of the lecture is "Time ordering and clock synchronization". This lecture covered the following topics.

Clock Synchronization : Cristians algorithm, Berkeley algorithm, NTP, GPS

Logical Clocks : Event Ordering

12.2 Clock Synchronization

12.2.1 The motivation of clock synchronization

In centralized systems and applications, it is not necessary to do clock synchronization since all entities use system clock of one machine for time and you can determine the order of sequence that events take place according to their local timestamp.

However, in distributed system, it may cause problems without clock synchronization. It is because each machine has its own system clock in distributed system, and one may run faster than the other without clock synchronization, thus you cannot determine event A in one machine occurs before event B in another machine only according to their local timestamp. For example, you modify files and save them on one machine A, and use another machine B to compile the files modified. If machine B has a faster clock than machine A, you may not correctly compile the files modified because the time of compiling files on machine B may be later than the time of editing files on machine A according to local timestamp on different machines, thus leading to errors.

12.2.2 How physical clocks and time work

1) Use astronomical metrics (solar day) to tell time: solar noon is the time that sun is directly overhead. noon is different with solar noon. noon depends on time zone while solar noon is exactly the same time. We typically use the notion of solar day to tell there are 24 hours between the time that sun is directly overhead on a particular location. Although this method is used for century, it is not accurate since it based on the length of a day.

2) Accurate clocks are atomic oscillators: It uses the property of atoms. the accuracy is one part in 1013. It is the most accurate clock used today. You have an atomic clock and broadcast clock values, then the receivers are synchronized with the atomic clock. For example, in US, there is a shortwave radio station that broadcast the atomic clock value, so that the receiver this and listening. cell-phone clock also uses atomic

clock to synchronized time with cell-phone broadcast tower. Some satellites also broadcast atomic clock.

3) Coordinated universal time(UTC): International standard based on atomic time. UTC broadcast on radio and receivers accurate to 0.1-10ms.

4) Mechanical clock: less accurate and the accuracy is one part in million. Computers use mechanical clock. It would result in clock drift since the property of quartz would change with the change of temperature humidity. To avoid clock drift, we need to synchronize machines with a master or with one another.

12.2.3 Drift tolerance and frequency of synchronization

If t is for UTC time and C is for clock time, then dC/dt equals to 1 for perfect clock. 1s in UTC time corresponds to 1s increase in clock time. While dC/dt is less than 1 for slow clock (the clock gains 1- ρ when 1s increases in UTC time) and larger than 1 for fast clock.

If there are two machines, each clock has a maximum drift rate ρ, then 1 - $\rho \leq dC/dt \leq 1+\rho$. Hence, the two clocks in two machines may drift by 2ρ in time in the worst case (one has a fast clock while the other has a slow clock). If the systems are to limit time drift to δ, we need to resynchronize every $\delta/2\rho$ seconds.

12.2.4 Centralized clock synchronization algorithms

12.2.4.1 Cristians algorithm

It assumes there is a master machine called time server, which somehow synchronizes with an atomic clock via a UTC receiver and has a timestamp for the system. Other machines of the system synchronize with the time server. Every $\delta/2\rho$ seconds, machine P sends network messages to time server to check what is the current time, and time server returns the current time and machine P uses the time to reset the clock of the machine P.

For example, machine P request time from time server. After treq time server receives request and returns time t to machine P. After treply machine P receives the time t from server and sets clock to t + t_{reply}, not t since we need to take the network propagation delay into account. We can use (t_{req} + t_{reply})/2 = (the time of machine P receives reply the time of machine P sends request) / 2 as an estimation of t_{reply}. In such estimation, you assume the time from machine P to server is identical to the time from server to machine P though it is usually not the case in practice, but we can use this assumption to give an estimation. To improve accuracy, we can estimate treply by making a series of measurements.

12.2.4.2 Berkeley algorithm

This algorithm just keeps clocks synchronized with one another in a group, and no machine in this group synchronize with external atomic clock. For this algorithm, the absolute time value is not important and we want to know clock differences between machines in a certain system. In this algorithm, we use leader election to select a master in a group to run clock synchronization while others are slaves. The way to do clock synchronization can be done in the following steps: a) Master asks all the other machines for their clock values by sending the time value on master; b) The other machines answer; c) Master tells everyone how to adjust their clock. Since some clock are fast and some clock are slow, this algorithm takes average time

difference to adjust their clock. For example, three machines reply with their clock values as time difference of 0, -10, +25 at 3:00, then the master will tell all those machines to set their clock at 3:00 + 5 (5 = (0 - 10 + 25) / 3). It is a relative clock synchronization, not absolute synchronization.

12.2.5 Distributed clock synchronization approaches

Cristians algorithm and Berkeley algorithm are both centralized approaches since they need time sever or time demon that run clock synchronization.

There are also decentralized algorithms using resynchronized intervals. Each machine in a certain system broadcasts time at the start of the interval and collects all other broadcast that arrive in a period S, then uses average value of all reported times. For the outliers, machines can throw away few highest and lowest values to avoid negative influence of extremely fast or slow clocks to the average time.

There are two decentralized approaches in use today. One approach is using NTP which is used in most computers to synchronize clock. NTP uses advanced techniques for accuracies of 1-50ms. If the time difference in the system is smaller than accuracy of NTP clock synchronization, then you cannot use NTP since you cannot say which event happens before another. The other approach is rdate which synchronizes a machine with a specified machine. In many cases, you can run rdate with the argument of the name of server and just synchronize clock with that server.

12.2.5.1 Network Time Protocol(NTP)

NTP is widely used standard which based on Cristians algorithm. In NTP clock synchronization, you also want to find out network propagation delay (dT_{res}). The difference between NTP and Cristians algorithm is NTP clock synchronization uses hierarchical protocol and does not let the clock to be set backward. For Cristians algorithm, fast clock and slow clock can just resynchronize with time server to adjust their time. However, for NTP clock synchronization, fast clock cannot go backward, and it is synchronized by slowing down fast clock. Letting clock go backward can cause many negative consequences (Like two files having the same timestamp). This is the reason why NTP is widely used compared to Cristians algorithm.

12.2.5.2 Global Positioning System(GPS)

GPS not only does clock synchronization, but also figures out the location of the block and the location of the receiver. GPS achieves high accuracy because it is synchronized with satellites which use atomic clock without hierarchical protocol.

GPS landmarks broadcast their locations and an unknown node will estimate its location according to landmarks locations.

How GPS works: We assume GPS landmark A with its position (x1, y1, z1) and its timestamp t1, and GPS receiver B (e.g. a car) with its unknown position (x, y, z) and the timestamp t receiving broadcast t1 message from GPS landmark. Then the distance between A and B is di = sqrt[(x1 - x)^2 + (y1 - y)^2 + (z1 - z)^2], and di also equals to c(t t1), c is the speed of light. We assume the receiver has a drift time dr from landmark A, then we can conduct the equation (1) that c(t + dr - t1) = sqrt[(x1 - x)^2 + (y1 - y)^2 + (z1 - z)^2]. From the equation (1), we can see that there are 4 unknowns x, y, z and dr, thus we need minimum 4 satellites to compute the location of a GPS receiver as well as its time value. If we get 4 satellites, then we can get multiple solutions of the location of the receiver. If we have 6 or 8 satellites, we can quickly narrow

the solutions. Therefore, GPS does clock synchronization as well as computing receivers location.

12.3 Logical clock

The above clock synchronization approaches assume synchronize local clock and use timestamp to reason the order of events. If the time difference between two events is smaller than the accuracy, then we cannot say which event happens first, thus problems may be caused. For many problems, we care about the internal consistency of clocks, not absolute time. The approach using logical clocks is proposed. Logical clocks do not need clock synchronization and take the order in which events occur rather than the time at which they occurred into account. If the processes only care about event A happens before event B, but dont care about the time difference exactly, they can use logical clock.

12.3.1 Event Ordering

Logical clocks are to solve the problem that define a total ordering of all events that occur in a system. In a distributed system, logical clocks do not have global clock and local clocks may be unsynchronized. Besides, you cannot order events on different machines using local times. There are some key ideas of logical clocks proposed by scientist Lamport: can use send/receive messages exchanged between processes/machines to order events since messages must be sent before received.

For example, machine A sends a message to machine B with its logical clock value 4, and machine B receives this message at its local logical clock value 3, then we can say that the events happens before logical clock value 4 in machine A occur before the events happens after logical clock value 3 in machine B without clock synchronization. However, we cannot say the order of the events happens after logical clock value 4 in machine A and the events happens before logical clock value 3 in machine B. If machine B receives a message at its local clock LCj from machine A with As logical clock LCi, then machine B would set its logical clock with logical clock of maximum(LCj + LCi) + 1.